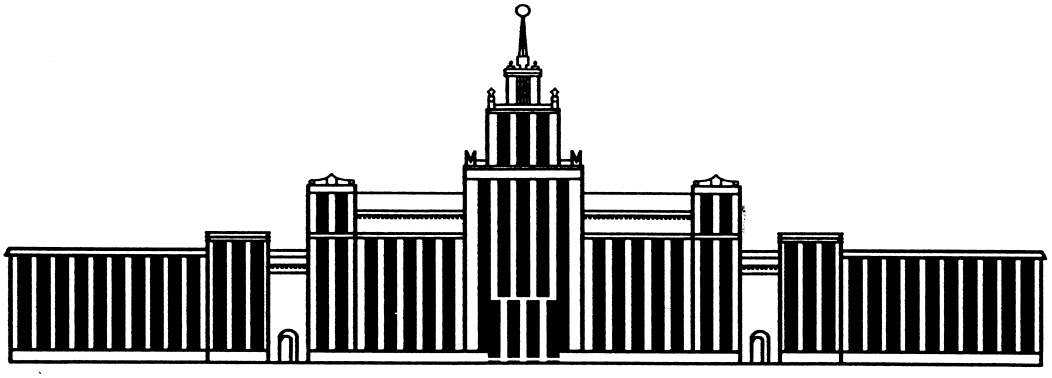
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ



ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

004(07)
Е552

С. М. Елсаков

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

Учебное пособие

Челябинск
2014

УДК 511(075.8)+[004.056.5:511](075.8)
Е552

Одобрено учебно-методической комиссией факультета математики, механики и компьютерных наук

Рецензенты:

Кораблева В. В., д.ф.-м.н., профессор кафедры компьютерной безопасности и прикладной алгебры ЧелГУ, Нигматулин Р. М., к.ф.-м.н., доцент кафедры математики и МОМ ЧГПУ

Елсаков, С. М.

Е552 Математические методы защиты информации: учебное пособие / С. М. Елсаков. — Челябинск: Издательский центр ЮУрГУ, 2014. — 142 с.

Учебное пособие предназначено для студентов очной формы обучения по направлениям 010400.62 «Прикладная математика и информатика», 231300.62 «Прикладная математика», 231000.62 «Программная инженерия». В пособии рассмотрены общие подходы к защите информации и математические основы криптографических алгоритмов. Приведены начальные сведения из теории групп, теории чисел, необходимые для первоначального знакомства с криптографическими алгоритмами. Содержатся материалы для проведения практических занятий, лабораторных и контрольных работ по курсу «Математические основы защиты информации».

УДК 511(075.8)+[004.056.5:511](075.8)

Министерство образования и науки Российской Федерации

Южно-Уральский государственный университет

Кафедра «Прикладная математика»

004(07)
Е552

С. М. Елсаков

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

Учебное пособие

Челябинск

Издательский центр ЮУрГУ

2014

1. ЗАЩИТА ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

Активное проникновение информационных технологий во все аспекты жизни современного общества выводит на первый план вопросы, связанные с безопасностью. Большое влияние на темпы развития информационных систем оказывает бурный рост производительности вычислительных систем, широкое распространение мобильных вычислительных систем, а также открывающиеся возможности при использовании информационных систем.

В рамках курса «Математические основы защиты информации» в первую очередь будут рассмотрены вопросы, связанные именно с защитой информации. Особенностью информации является ее неразрывная связь с материальным носителем. Носителем может выступать бумага (информация записана карандашом), память устройств (уровень напряжения на определенной шине или уровень напряженности магнитного поля), радиоканал (информация закодирована с помощью модуляции в параметрах сигнала), оптический канал связи (информация закодирована с помощью фотонов) и т. д. В дальнейшем будем предполагать, что информация обрабатывается в рамках некоторой информационной системы.

Определение 1. *Информационной системой (ИС)* называется совокупность средств, методов и персонала, обеспечивающих сбор, хранение, обработку, передачу и отображение информации в интересах достижения поставленной цели.

Информационные системы предназначены для удовлетворения потребностей пользователей в получении информации. Информационные процессы протекающие в ИС определяются соответствующей информационной технологией.

Определение 2. *Информационная технология* — это совокупность средств и методов сбора, обработки, передачи данных для получения информации нового качества о состоянии объекта, процесса или явления.

Примером информационной системы может являться бухгалтерский учет в любой организации. Бухгалтерия в качестве информационной технологии может использовать как «бумажную» технологию, широко используемую в XX веке, так и «электронную» на основе Microsoft Excel для малых предприятий (локальный программный продукт), 1С:Предприятие для средних предприятий (клиент-серверный программный про-

дукт), SAP для крупных предприятий (трехзвеневая модель), а также различные облачные решения.

Информационные системы представляют собой совокупность взаимосвязанных компонентов:

1. Информационных массивов (все возможные виды памяти);

2. Технических средств обработки и передачи данных (сети передачи);

3. Программных средств (алгоритмическая основа информационной системы);

4. Персонала и пользователей системы.

Защита информации предусматривает комплексное использование различных средств и методов, принятие соответствующих мер, осуществление необходимых мероприятий с целью системного обеспечения надежности передаваемой, хранимой и обрабатываемой информации.

Рассмотрим следующий пример. Пусть перед нами стоит задача разработки программного продукта, обеспечивающего всех жителей Российской Федерации единой медицинской электронной картой. С помощью этой информационной системы мы должны каждому гражданину РФ обеспечить индивидуальную медицинскую карту, врачам осуществляющим прием — возможность заполнять эту карточку. Безусловно в такой системе возникают ограничения на использование данных, поскольку часть информации является врачебной тайной, часть информации является персональными данными граждан. Кроме того, необходимо поддерживать конфиденциальность в отношении и самих лечебных учреждений, поскольку они могут быть коммерческими, и эти данные будут являются коммерческой тайной. Таким образом, в отношении информации мы хотим обеспечить выполнение следующих требований:

1. Конфиденциальность (данные должны получить только те субъекты, которые имеют на это право).

2. Доступность (данные карточек должны быть доступны оперативно, например, при оказании гражданину неотложной помощи).

3. Целостность (данные должны храниться в неизменном виде в течение всей жизни пациента).

4. Неотказуемость (данные, внесенные врачом, должны быть защищены от отказа этого врача признать их своими).

5. Аутентичность (система должна подтверждать, что эти данные внесены именно этим врачом)

Рассмотрим, какие методы могут применяться для обеспечения защиты информации.

1.1. Общая характеристика средств и методов защиты информации в компьютерных системах

Противодействие многочисленным угрозам информационной безопасности ИС предусматривает комплексное использование различных способов и мероприятий организационного, правового, инженерно–технического, программно–аппаратного, криптографического характера.

Правовые нормы и организационные методы защиты информации включают меры, мероприятия и действия, которые должны осуществлять должностные лица в процессе создания и эксплуатации ИС для обеспечения заданного уровня безопасности информации. Основу обеспечения информационной безопасности составляют различные нормативно–правовые акты, требования которых обязательны в системе защиты информации. Только с помощью организационных методов возможно объединение на правовой основе технических, программных и криптографических средств защиты информации в единую комплексную систему.

К инженерно–техническим средствам защиты относятся те, которые создают физически замкнутую среду вокруг элементов защиты, создавая тем самым определённое препятствие для злоумышленника. Для этой цели применяют различные инженерные конструкции, обеспечивающие охрану территории и помещений, используют автономные средства защиты аппаратуры с контролем их возможного вскрытия, применяют устройства с низким электромагнитным и акустическим излучением, осуществляют экранирование помещений, обеспечивают энергоснабжение от автономного источника электропитания и т.д.

Программно-аппаратные средства защиты непосредственно применяются в компьютерах и компьютерных сетях, содержат различные встраиваемые в ИС устройства, а также специальные пакеты программ или отдельные программы, реализующие такие функции защиты как разграничение и контроль доступа к ресурсам, регистрация и анализ протекающих процессов, событий, пользователей, предотвращение возможных разрушительных воздействий на ресурсы, идентификация и аутентификация пользователей и процессов и др.

Существенное повышение информационной безопасности, особенно при передаче данных в компьютерных сетях или при их обмене между удалёнными объектами, достигается применением средств криптографической защиты. Суть криптографической защиты заключается в преобразовании информации к неявному виду с помощью специальных алгоритмов.

1.2. Основные принципы защиты информации

При защите информации необходимо соблюдать следующие основные принципы.

Принцип системности. Системный подход к защите ИС предполагает необходимость учёта всех взаимосвязанных, взаимодействующих и изменяющихся во времени элементов, условий и факторов: для всех видов информационной деятельности и информационного проявления, во всех структурных элементах, при всех режимах функционирования, на всех этапах жизненного цикла ИС, с учётом взаимодействия объекта защиты с внешней средой.

Принцип комплексности. Комплексное использование широкого спектра различных методов и средств защиты информации предполагает согласование разнородных средств при построении целостной системы защиты, перекрывающей все существующие, а также возможные каналы реализации угроз и не содержащие слабых мест на стыках отдельных её компонентов.

Рассмотрим некоторую информационную систему. Мы можем сформулировать ряд угроз, свойственных этой системе, рис. 1.1. Для каждой угрозы мы можем указать на какие компоненты информационной системы каждая угроза может распространяться. Таким образом, у нас сформировался двудольный граф: вершины первой доли соответствуют угрозам, вершины второй доли — компонентам информационной системы, ребра соединяют компоненты и угрозы, которые на них распространяются. Задача построения системы защиты заключается в перекрытии всех ребер, нейтрализации всех угроз или защите всех компонент ИС. Если у нас имеются численные оценки вероятности возникновения угроз (например, уже накоплен какой-то опыт), то задавая вероятности распространения угрозы на компоненты системы, мы можем получить численные оценки защищенности ИС.

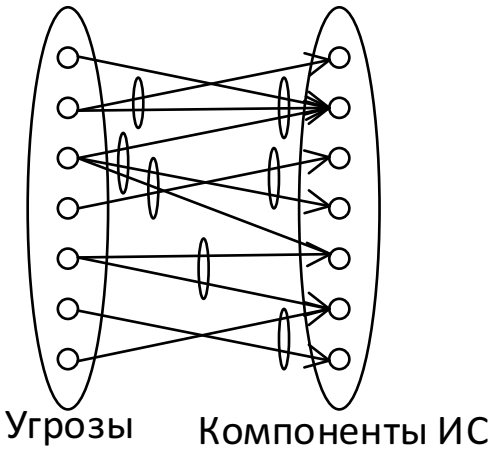


Рис. 1.1. Модель системы безопасности с полным перекрытием

Принцип непрерывности защиты. Защита информации – это непрерывный целенаправленный процесс, предполагающий принятие соответствующих мер на всех этапах жизненного цикла ИС (начиная с самых ранних стадий проектирования). Разработка системы защиты

должна вестись параллельно с разработкой защищаемой компьютерной системы. Только в этом случае возможно эффективно обеспечить реализацию всех остальных принципов.

Непрерывность защиты должна быть обеспечена при использовании, в частности, различных технологий защиты информации. Когда эти технологии дополняют друг друга, то можно говорить о модели многоаспектной защиты информации (рис. 1.2). В рамках этой модели информация помещается в центр, а вокруг нее возводится некая преграда для злоумышленника. Сложность преодоления этой преграды злоумышленником называется прочностью преграды. Предполагается, что ценность информации, помещенной за преграду, ниже, чем стоимость взлома преграды. В случае оценки информации в деньгах, это означает, что стоимость затраченных денег на взлом преграды существенно больше, чем предполагаемая стоимость информации. Если информация имеет небольшой срок актуальности, то от прочности преграды можно требовать, чтобы на взлом преграды времени уходило бы больше, чем срок актуальности информации. Если преграда состыкована из различных участков, то с учетом того, что нарушитель всегда выбирает самую «слабую» преграду, а за укрепление преграды всегда приходится расплачиваться владельцу информации, то разумным представляется подход, когда прочность всегда поддерживается на примерно одном выбранном уровне, соответствующем ценности информации.

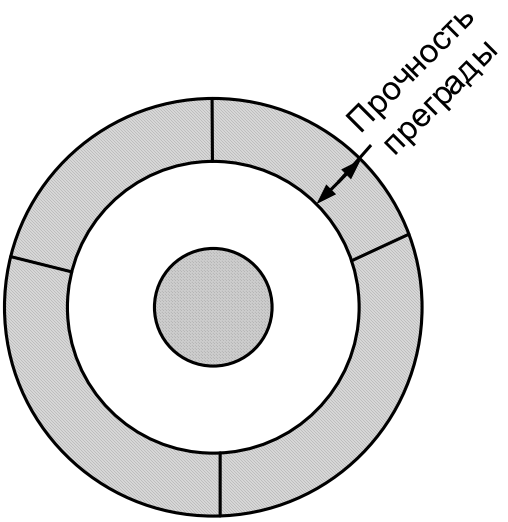


Рис. 1.2. Модель многоаспектной защиты информации

Разумная достаточность. Создать абсолютно непреодолимую систему защиты принципиально невозможно. При достаточных средствах и времени можно преодолеть любую защиту. Поэтому имеет смысл вести речь только о некотором приемлемом уровне безопасности. Высокоэффективная система защиты стоит дорого, использует при работе существенную часть ресурсов компьютерной системы и может создавать ощутимые дополнительные неудобства для пользователей. Важно правильно выбрать тот достаточный уровень защиты, при котором затраты, риск и размер возможного ущерба были бы приемлемыми.

Одним из стандартных способов повышения надежности защиты является метод многоуровневой защиты, когда для выполнения одного действия пользователя требуется пройти несколько разных уровней защиты (рис. 1.3). Например, для пользования чиповой банковской картой необходимо иметь саму карточку и знать пин-код.

Гибкость системы защиты. Часто приходится создавать системы защиты в условиях большой неопределённости. Кроме того, внешние условия и требования с течением времени меняются. Поэтому принятые меры и установленные средства защиты, особенно в начальный период их эксплуатации, могут обеспечить как чрезмерный, так и недостаточный уровень защиты. Естественно, для обеспечения возможности коррекции этого уровня средства защиты должны обладать определённой гибкостью.

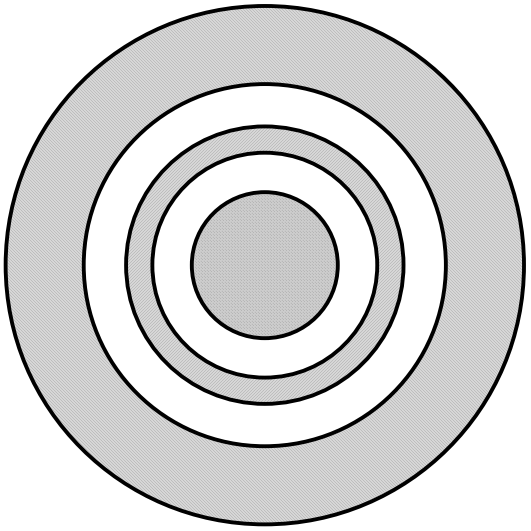


Рис. 1.3. Модель многоуровневой защиты информации

Открытость алгоритмов и механизмов защиты. Суть этого принципа состоит в том, что защита информации не должна обеспечиваться только за счёт секретности структурной и функциональной организации системы защиты. Специалисты, имеющие отношение к системе защиты, должны полностью представлять себе принципы её функционирования и в случае возникновения затруднительных ситуаций адекватно на них реагировать. Однако это вовсе не означает, что информация о конкретной системе защиты должна быть общедоступна – необходимо обеспечивать защиту от угрозы раскрытия параметров системы.

Принцип простоты применения средств защиты. Механизмы защиты должны быть интуитивно понятны и просты в использовании. Защита будет тем более эффективной, чем легче пользователю с ней работать. Применение средств защиты не должно быть связано со знанием специальных языков или с выполнением действий, требующих значительных дополнительных трудовых затрат при обычной работе законных пользователей.

2. КЛАССИЧЕСКИЕ ШИФРЫ

Проблема защиты информации путем ее преобразования, исключающего ее прочтение посторонним лицом, волновала человечество с давних времен. История криптографии — ровесница истории человеческого языка. Более того, первоначально письменность сама по себе была своеобразной криптографической системой, так как в древних обществах ею владели только избранные. В истории криптографии можно выделить несколько этапов.

Для наивной криптографии (до нач. XVI века) характерно использование способов запутывания противника относительно содержания шифруемых текстов. На начальном этапе для защиты информации использовались методы кодирования и стеганографии (см. стр. 120). Большинство из используемых шифров сводились к шифрам перестановки или моноалфавитной подстановке.

Одним из первых зафиксированных примеров является шифр Цезаря, состоящий в замене каждой буквы исходного текста на другую, отстоящую от нее в алфавите на определенное число позиций. Для описания шифров введем следующее определение.

Определение 3. *Стандартное кодирование* букв алфавита — это последовательное сопоставление буквам алфавита чисел от 1 до последней буквы алфавита. Нулем кодируется символ пробел.

_	A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Рис. 2.1. Стандартное кодирование букв английского алфавита

_	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

Рис. 2.2. Стандартное кодирование букв русского алфавита

В дальнейшем мы будем неоднократно возвращаться к стандартному кодированию букв алфавита (рис. 2.1 и рис. 2.2). Например, с помощью стандартного кодирования удобно записать формулу для шифрования по схеме Цезаря:

$$C_i = (M_i + K) \bmod L,$$

где C_i — i-й символ шифротекста (символ алфавита), M_i — i-й символ входного сообщения (символ алфавита), K — ключ (символ алфавита), L — размер алфавита (27 для английского языка с учётом пробела).

Дешифрование выполняется по формуле

$$M_i = (C_i - K) \bmod L.$$

Полибианский квадрат(Полибий, 200–120 д. н. э.) является общей моноалфавитной подстановкой, которая проводится с помощью случайно заполненной алфавитом квадратной таблицы. Каждая буква исходного текста заменяется на букву, стоящую в квадрате снизу от нее рис. 2.3. Буквы последней строки заменяются на буквы первой строки.

С	Ь	Ж	Н	Ф	Ъ	К
Ц	Б	Щ	Я	Р	Д	И
Ш	_	Т	М	А	Ч	Ы
Г	Х	З	,	П	О	Ю
Л	Й	.	Е	Э	В	У

Рис. 2.3. Полибианский квадрат

Метод шифрования, называемый одиночной перестановкой по ключу, использует перестановку букв исходного текс. Текст записывается в некоторую таблицу по столбацам, затем столбцы таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы, а затем текст построчно выписывается как шифротекст. Ключом является ключевое слово и размер таблицы. Применим в качестве ключа, например, слово ПЕЛИКАН.

П	Е	Л	И	К	А	Н
7	2	5	3	4	1	6
Т	Н	П	В	Е	Г	Л
Е	А	Р	А	Д	О	Н
Р	Т	И	Е	Ь	В	О
М	О	Б	Т	М	П	Ч
И	Р	Ы	С	О	О	Ь

До перестановки

А	Е	И	К	Л	Н	П
1	2	3	4	5	6	7
Г	Н	В	Е	П	Л	Т
О	А	А	Д	Р	Н	Е
В	Т	Е	Ь	И	О	Р
П	О	Т	М	Б	Ч	М
О	Р	С	О	Ы	Ь	И

После перестановки

Рис. 2.4. Одиночная перестановка по ключу

На рис. 2.4 показаны две таблицы: левая таблица соответствует заполнению до перестановки, а правая таблица — заполнению после перестановки. В верхней строке левой таблицы записан ключ, а номера под буквами ключа определены в соответствии с порядком соответствующих букв ключа в алфавите. Если бы в ключе встретились одинаковые буквы, они бы были занумерованы слева направо. В правой таблице столбцы переставлены в соответствии с упорядоченными номерами букв ключа. При считывании содержимого правой таблицы по строкам и записи шифротекста блоками по пять букв получим шифрованное сообщение: ГНВЕП ЛТООА ДРНЕВ ТЕЬИО РПОТМ БЧМОР СОЫЬИ. Для обеспечения дополнительной скрытности можно повторно зашифровать сообщение, которое уже прошло шифрование. Такой метод шифрования называется двойной перестановкой. В случае двойной перестановки столбцов и строк таблицы перестановки определяются отдельно для столбцов и отдельно для строк. Сначала в таблицу записывается текст сообщения, а потом поочередно переставляются столбцы, а затем строки. При дешифровании порядок перестановок должен быть обратным. Пример выполнения шифрования методом двойной перестановки показан на рис. 2.5.

	С(6)	К(3)	А(1)	Н(4)	Е(2)	Р(5)
К(1)	С	И	С	Т	Е	М
Р(2)	Н	Ы	Й	_	П	А
Ы(3)	Р	О	Л	Ь	_	И
М(4)	З	М	Е	Н	Е	Н

Исходная таблица

	А(1)	Е(2)	К(3)	Н(4)	Р(5)	С(6)
К(1)	С	Е	И	Т	М	С
Р(2)	Й	П	Ы	_	А	Н
Ы(3)	Л	_	О	Ь	И	Р
М(4)	Е	Е	М	Н	Н	З

Перестановка столбцов

	А(1)	Е(2)	К(3)	Н(4)	Р(5)	С(6)
К(1)	С	Е	И	Т	М	С
М(2)	Е	Е	М	Н	Н	З
Р(3)	Й	П	Ы	_	А	Н
Ы(4)	Л	_	О	Ь	И	Р

Перестановка строк

Рис. 2.5. Двойная перестановка по ключу

Если считывать шифротекст из правой таблицы построчно блоками по шесть букв, то получится следующее: СЕИТМС ЕЕМННЗ ЙПЫ_АН Л_ОЬИР. Ключом к шифру двойной перестановки служит последовательность номеров столбцов и номеров строк исходной таблицы (в нашем примере последовательности 631425 и 1342 соответственно).

В средние века для шифрования перестановкой применялись магические квадраты. Магическими квадратами (рис. 2.6) называют квадрат-

ные таблицы с вписанными в их клетки последовательными натуральными числами, начиная от 1, которые дают в сумме по каждому столбцу, каждой строке и каждой диагонали одно и то же число. Исходный текст вписывали в магические квадраты в соответствии с нумерацией их клеток. Если затем выписать содержимое такой таблицы по строкам, то получится шифротекст, сформированный благодаря перестановке букв исходного сообщения. Число магических квадратов быстро возрастает с увеличением размера квадрата. Существует только один магический квадрат размером 3×3 (если не учитывать его повороты). Количество магических квадратов 4×4 составляет уже 880, а количество магических квадратов 5×5 — около 250000.

Правители Спарты шифровали свои послания с помощью скитала, первого простейшего криптографического устройства, реализующего метод простой перестановки следующим образом. На стержень цилиндрической формы, который назывался скитала, наматывали спиралью (виток к витку) полоску пергамента и писали на ней вдоль стержня несколько строк текста сообщения. Затем снимали со стержня полоску пергамента с написанным текстом. Буквы на этой полоске оказывались расположенными хаотично. Сообщение НАСТУПАЙТЕ при размещении его по окружности стержня по три буквы дает шифротекст НУТАПЕСА_ТЙ Для расшифрования такого шифртекста нужно не только знать правило шифрования, но и обладать ключом в виде стержня определенного диаметра. Зная только вид шифра, но не имея ключа, расшифровать сообщение было непросто.

4	9	2
3	5	7
8	1	6

3×3

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

4×4

Рис. 2.6. Магические квадраты

Этап формальной криптографии (кон. XV века — нач. XX века) связан с появлением формализованных и относительно стойких к ручному криптоанализу шифров. В европейских странах это произошло в эпоху Возрождения, когда развитие науки и торговли вызвало спрос на надежные способы защиты информации.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	
A	A	B	C	D	E	F	G	H	
B	Z	A	B	C	D	E	F	G	
C	Y	Z	A	B	C	D	E	F	
D	X	Y	Z	A	B	C	D	E	...
E	W	X	Y	Z	A	B	C	D	
F	V	W	X	Y	Z	A	B	C	
G	U	V	W	X	Y	Z	A	B	
H	T	U	V	W	X	Y	Z	A	

...

Рис. 2.7. Шифр Виженера

Наибольшую известность получил шифр, названный в

честь Б. Виженера (XVI век), который состоял в последовательном «сложении» букв исходного текста с ключом (процедуру можно ускорить с помощью специальной таблицы, рис. 2.7):

$$C_i = (M_i + K_{i \bmod L_k}) \bmod L,$$

где C_i — i-й символ шифротекста (символ алфавита), M_i — i-й символ входного сообщения (символ алфавита), K_i — i-й символ ключевой фразы (символ алфавита), L — размер алфавита (27 для английского языка с учётом пробела), L_k — длина ключевой фразы.

Дешифрование выполняется по формуле

$$M_i = (C_i - K_{i \bmod L_k}) \bmod L.$$

Одной из первых печатных работ, в которой обобщены и сформулированы известные на тот момент алгоритмы шифрования является «Полиграфия» (1508 г.) И. Трисемуса. Ему принадлежат два небольших, но важных открытия: способ заполнения полибианского квадрата (первые позиции заполняются с помощью легко запоминаемого ключевого слова, остальные — оставшимися буквами алфавита, см. рис. 2.8) и шифрование пар букв (биграмм).

П	А	Р	О	Л	Ь
Б	В	Г	Д	Е	Ё
Ж	З	И	Й	К	М
Н	С	Т	У	Ф	Х
Ц	Ч	Ш	Щ	Ъ	Ы
Э	Ю	Я	_	.	,

Рис. 2.8. Легко запоминаемая таблица Трисемуса

Стойким способом полиалфавитной замены является шифр Плейфера, который был переоткрыт в начале XIX века Ч. Уитстоном. Уитстону принадлежит и важное усовершенствование — шифрование «двойным квадратом». Шифры Плейфера (рис. 2.9) и Уитстона (рис. 2.10) использовались вплоть до первой мировой войны, так как с трудом поддавались ручному криптоанализу.

Шифр Плейфера заключается в разбиении всего текста на биграммы, которые шифруются с помощью таблицы независимо друг от друга. Для шифрования биграммы необходимо найти буквы, составляющие биграмму, в таблице, мысленно построить прямоугольник так, чтобы эти буквы оказались в противоположных вершинах этого прямоугольника. Широтекстом будет пара букв расположенных в оставшихся противоположных вершинах (рис. 2.9). Если буквы биграммы расположены в одной строке или в одном столбце, то необходимо их заменить на буквы расположенные правее или ниже соответственно (рис. 2.9). Шифр Уитстона аналогичен шифру Плейфера за исключением того, что в шифре

Уитстона буквы, составляющие биграмму, необходимо искать в разных таблицах (рис. 2.10).

5	9	6	8	7	4
Z	3	0	2	1	Y
N	R	O	**Q**	P	M
B	F	C	E	D	A
T	X	U	W	V	S
H	L	I	K	J	G

NQ → RP

5	9	6	8	7	4
Z	3	0	2	1	Y
N	R	O	Q	P	M
B	F	C	E	D	A
T	X	U	W	V	S
H	L	I	K	J	G

ZT → NH

5	9	6	8	7	4
Z	3	0	2	1	Y
N	R	O	Q	P	M
B	**F**	C	E	D	A
T	X	U	W	V	S
H	L	I	K	**J**	G

FJ → DL

Рис. 2.9. Шифр Плейфера

5	9	6	8	7	4
Z	3	0	2	1	Y
N	R	O	Q	P	M
B	F	C	E	D	A
T	X	U	W	V	S
H	L	I	K	J	G

X	S	W	T	U	V
F	A	E	B	C	D
9	4	8	5	6	7
L	G	K	H	I	J
3	Y	2	Z	0	1
R	M	Q	**N**	O	P

ZN → BH

Рис. 2.10. Двойной квадрат Уитстона

Система омофонов обеспечивает простейшую защиту от криптоаналитических атак, основанных на подсчете частот появления букв в шифртексте. В системе омофонов буквы исходного сообщения имеют несколько замен. Число замен берется пропорциональным вероятности появления буквы в открытом тексте. Шифруя букву исходного сообщения, выбирают случайным образом одну из ее замен. Замены (часто называемые омофонами) могут быть представлены трехразрядными числами от 000 до 999. Например, в английском алфавите букве E присваиваются 123 случайных номера, буквам B и G — по 16 номеров, а буквам J и Z — по 1 номеру. Если омофоны (замены) присваиваются случайным образом различным появлениям одной и той же буквы, тогда каждый омофон появляется в шифртексте равновероятно.

Шифр Хилла — это полиалфавитный шифр подстановки, основанный на линейной алгебре. Л. Хилл предложил этот шифр в 1929 г. Каждой букве сперва сопоставляется число. Для латинского алфавита часто используется простейшая схема: $A = 0, B = 1, ..., Z = 25$, но это не является существенным свойством шифра. Блок из n букв рассматривается как n-мерный вектор, который при шифровании умножается на $n \times n$ матрицу по модулю 26. Матрица целиком является ключом шифра. Матрица должна быть обратима над $\mathbb{Z}_{26}$, чтобы была возможна операция расшифрования. Рассмотрим сообщение «DOG» и представленный

ниже ключ (GYBNQKURP в буквенном виде):

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}.$$

Так как букве «D» соответствует число 3, «O» — 14, «G» — 6, то сообщение — это вектор $\begin{pmatrix} 3 & 14 & 6 \end{pmatrix}^T$. Тогда зашифрованный вектор будет равен

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 3 \\ 14 \\ 6 \end{pmatrix} \equiv \begin{pmatrix} 360 \\ 323 \\ 388 \end{pmatrix} \equiv \begin{pmatrix} 22 \\ 11 \\ 24 \end{pmatrix} \pmod{26}$$

что соответствует шифротексту «WLY». Теперь предположим, что наше сообщение было «GOD» или $\begin{pmatrix} 6 & 14 & 3 \end{pmatrix}^T$. Теперь зашифрованный вектор будет

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 6 \\ 14 \\ 3 \end{pmatrix} \equiv \begin{pmatrix} 375 \\ 332 \\ 403 \end{pmatrix} \equiv \begin{pmatrix} 11 \\ 20 \\ 13 \end{pmatrix} \pmod{26},$$

что соответствует шифротексту «LUN». Видно, что каждая буква шифротекста сменилась. Для того, чтобы расшифровать сообщение, необходимо обратить шифротекст обратно в вектор и затем просто умножить на обратную матрицу ключа (IFKVIVVMI в буквенном виде). Обратная матрица над $\mathbb{Z}_{26}$, к использованной матрице в примере шифрования, будет равна

$$\begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}.$$

Возьмем шифротекст из предыдущего примера «WLY». Тогда мы получим

$$\begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 22 \\ 11 \\ 24 \end{pmatrix} \equiv \begin{pmatrix} 471 \\ 1054 \\ 786 \end{pmatrix} \equiv \begin{pmatrix} 3 \\ 14 \\ 6 \end{pmatrix} \pmod{26},$$

что возвращает нас к сообщению «DOG», как мы и рассчитывали. Необходимо учитывать, что не все матрицы имеют обратную. Матрица будет иметь обратную в том и только в том случае, когда ее детерминант не равен нулю и не имеет общих делителей с основанием модуля. Детерминант матрицы из примера:

$$\begin{vmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{vmatrix} \equiv 6(16 \cdot 15 - 10 \cdot 17) - 24(13 \cdot 15 - 10 \cdot 20) + \\ +1(13 \cdot 17 - 16 \cdot 20) \equiv 441 \equiv 25 \pmod{26}.$$

Итак, детерминант равен 25 по модулю 26. Так число 25 не имеет общих делителей с числом 26, то матрица с таким детерминантом может использоваться в шифре Хилла.

В XIX веке Керкхофф сформулировал главное требование к криптографическим системам, которое остается актуальным и поныне: секретность шифров должна быть основана на секретности ключа, но не алгоритма.

Последним словом в донаучной криптографии, которое обеспечили еще более высокую криптостойкость, а также позволило автоматизировать (в смысле механизировать) процесс шифрования стали роторные криптосистемы. Полиалфавитная подстановка с помощью роторной машины (рис. 2.11) реализуется вариацией взаимного положения вращающихся роторов, каждый из которых осуществляет «прошитую» в нем подстановку. Практическое распространение роторные машины получили только в начале XX века. Роторные машины активно использовались во время второй мировой войны (например, Enigma).

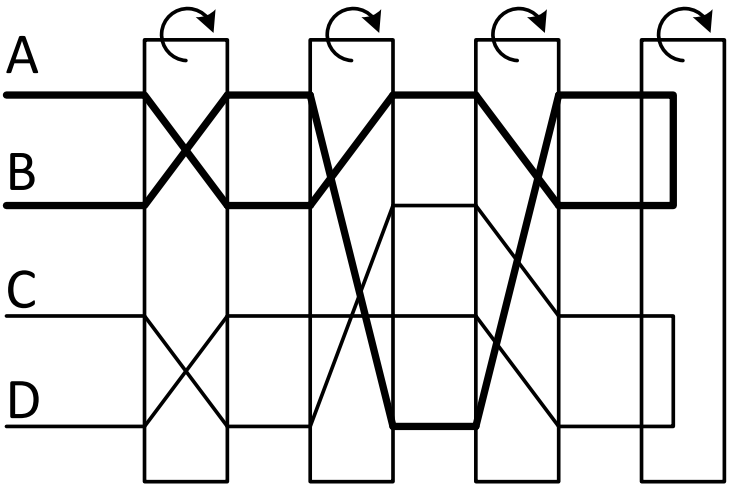


Рис. 2.11. Схема работы роторной машины

Главная отличительная черта научной криптографии (30-е — 60-е годы XX века) — появление криптосистем со строгим математическим обоснованием криптостойкости. Своеобразным водоразделом стала работа Клода Шеннона «Теория связи в секретных системах» (1949), где сформулированы теоретические принципы криптографической защиты информации. Шеннон ввел понятия «рассеивание» и «перемешивание», обосновал возможность создания сколь угодно стойких криптосистем.

Компьютерная криптография (с 70-х годов XX века) обязана своим появлением вычислительным средствам с производительностью, достаточной для реализации криптосистем, обеспечивающих при большой скорости шифрования на несколько порядков более высокую криптостойкость, чем «ручные» и «механические» шифры. В середине 70-х годов произошел настоящий прорыв в современной криптографии — появление асимметричных криптосистем, которые не требовали передачи секретного ключа между сторонами.

2.1. Криптостойкость

Главным действующим лицом в криптоанализе выступает нарушитель (или криптоаналитик). Под ним понимают лицо, целью которых является прочтение или подделка защищенных криптографическими методами сообщений. В отношении нарушителя принимается ряд допущений, которые как правило кладутся в основу математических или иных моделей:

1. Нарушитель знает алгоритм шифрования и особенности его реализации в конкретном случае, но не знает секретного ключа.

2. Нарушителю доступны все зашифрованные тексты. Нарушитель может иметь доступ к некоторым исходным текстам, для которых известны соответствующие им зашифрованные тексты.

3. Нарушитель имеет в своем распоряжении вычислительные, человеческие, временные и иные ресурсы, объем которых оправдан потенциальной ценностью информации, которая будет добыта в результате криптоанализа.

Попытку прочтения или подделки зашифрованного сообщения, вычисления ключа методами криптоанализа называют криптоатакой или атакой на шифр. Удачную криптоатаку называют взломом. Криптостойкостъю называется характеристика шифра, определяющая его стойкость к расшифрованию без знания ключа (т.е. криптоатаке).

Показатель криптостойкости — главный параметр любой криптосистемы. В качестве показателя криптостойкости можно выбрать: количество всех возможных ключей или вероятность подбора ключа за заданное время с заданными ресурсами; количество операций или время (с заданными ресурсами), необходимое для взлома шифра с заданной вероятностью; стоимость вычисления ключевой информации или исходного текста. Все эти показатели должны учитывать также уровень возможной криптоатаки. Однако следует понимать, что эффективность защиты информации криптографическими методами зависит не только от криптостойкости шифра, но и от множества других факторов, включая вопросы реализации криптосистем в виде устройств или программ. При анализе криптостойкости шифра необходимо учитывать и человеческий фактор. Например, подкуп конкретного человека, в руках которого сосредоточена необходимая информация, может стоить на несколько порядков дешевле, чем создание суперкомпьютера для взлома шифра.

1. При *атаке по шифрованному тексту* Нарушителю доступны все или некоторые зашифрованные сообщения.

2. При *атаке по паре «исходный текст — шифрованный текст»* Нарушителю доступны все или некоторые зашифрованные сообщения и соответствующие им исходные сообщения.

3. При *атаке по выбранной паре «исходный текст — шифрованный текст»* Нарушитель имеет возможность выбирать исходный текст, получать для него шифрованный текст и на основе анализа зависимостей между ними вычислять ключ. Все современные криптосистемы обладают достаточной стойкостью даже к атакам этого уровня, то есть когда нарушителю доступно по сути шифрующее устройство.

При обсуждении вопроса о степени безопасности криптосистемы могут быть использованы следующие два подхода.

Первый подход исходит от объема вычислений, требуемых для взлома криптосистемы. Криптосистема может быть названа вычислительно стойкой, если наилучший алгоритм для ее взлома требует по крайней мере N операций, где N — некоторое достаточно большое число. Проблема, однако, состоит в том, что ни для одной из используемых на практике криптосистем стойкость в смысле этого определения не доказана. На практике криптосистема считается вычислительно стойкой, если самый лучший из известных методов ее взлома требует неприемлемо большого объема вычислений (конечно, это весьма отличается от доказательства вычислительной стойкости). Иногда сводят вопрос о вычислительной стойкости к некоторой хорошо изученной задаче, которая считается трудной. Например, может быть доказано утверждение следующего типа: «данная криптосистема является стойкой, если данное натуральное число n не может быть разложено на простые множители». Криптосистемы этого типа иногда называются «доказуемо стойкими». Разумеется, это также далеко от доказательства вычислительной стойкости.

Второй подход заключается в том, что никаких ограничений на объем вычислений не делается. Криптосистема называется безусловно стойкой, если она не может быть взломана при использовании сколь угодно больших вычислительных ресурсов. Естественным аппаратом для изучения безусловной стойкости является теория вероятностей.

Мы будем рассматривать всюду далее дискретные случайные величины, принимающие конечное множество значений. Множество значений случайной величины X будем также обозначать через X. Что имеется в виду в данном случае будет ясно из контекста.

Пусть X и Y — случайные величины. Вероятность того, что величина X принимает значение x обозначим через $P(X = x)$. Совместную вероятность того, что величина X принимает значение x, а величина Y принимает значение y обозначим через $P(X = x, Y = y)$. Условная вероятность $P(X = x|Y = y)$ — это вероятность того, что случайная величина X принимает значение x при условии того, что случайная величина Y приняла значение y. Две случайные величины X и Y называются неза-

висимыми, если $P(X = x, Y = y) = P(X = x)P(Y = y)$ для любых их возможных значений.

Рассмотрим произвольную криптосистему (M, C, K), где M — множество исходных сообщений, C — множество шифротекстов, K — множество ключей. Обозначим вероятность того, что символ открытого текста принимает значение, равное m, через $p_M(m)$. Будем также предполагать, что ключ k выбирается некоторым случайным образом с известным распределением вероятностей. Обозначим вероятность выбора ключа k через $p_K(k)$. В силу описанной выше процедуры выбора ключей, естественно предполагать, что ключ k и открытый текст m являются независимыми случайными величинами.

Пусть функция шифрования это отображение $e(m, k) : M \times K \to C$, где $m \in M$, $k \in K$, $e(m, k) \in C$, а функция дешифрования — $d(c, k) : C \times K \to M$, где $c \in C$, $k \in K$, $d(c, k) \in M$.

Законы распределения в M и K однозначно определяют закон распределения в C. Найдем вероятность $p_C(c)$ того, что был передан шифротекст c. Для $k \in K$, обозначим:

$$E(k) = \{e(m, k) : m \in M)\}.$$

Иначе говоря, $E(k)$ — это множество всех возможных шифротекстов, получаемых с помощью шифрования с ключом k. Введем два вспомогательных множества $K_e(c) = \{k : e(m, k) = c, m \in M\}$ — множество всех ключей, которые могут использоваться, чтобы получить шифротекст c, и $K_d(c, m) = \{k : d(c, k) = m\}$ — множество всех ключей, позволяющих расшифровать шифротекст c в исходной сообщение m. Для произвольного $c \in C$ по формуле полной вероятности получаем

$$p_C(c) = \sum_{k \in K_e(c)} p_K(k) p_M\left(d(c, k)\right).$$

Для произвольных $c \in C$, $m \in M$ условная вероятность $p_C(c|m)$ (вероятность того, что шифротекстом будет c при условии, что открытым текстом является m) может быть найдена по формуле

$$p_C(c|m) = \sum_{k \in K_d(c,m)} p_K(k).$$

Теперь по формуле Байеса находим условную вероятность $p_M(m|c)$ (вероятность того, что открытым текстом будет m при условии, что шифртекстом является c):

$$p_M(m|c) = \frac{p_M(m) p_C(c|m)}{p_C(c)},$$

$$p_M(m|c) = \frac{p_M(m) \sum\limits_{k \in K_d(c,m)} p_K(k)}{\sum\limits_{k \in K_e(c)} p_K(k) p_M(d(c,k))}.$$

Заметим, что для проведения всех этих вычислений достаточно знать только распределения вероятностей.

Введем теперь понятие абсолютной стойкости криптосистемы. Это свойство означает, что нарушитель не может получить какой-либо информации об открытом тексте, зная шифротекст. Точная формулировка дается следующим определением.

Определение 4. Криптосистема называется *абсолютно стойкой*, если $p_M(m|c) = p_M(m)$ для любых $c \in C$, $m \in M$.

Иначе говоря, апостериорная вероятность того, что открытый текст совпадает с m при условии, что шифротекст совпадает с c, равна априорной вероятности того, что открытый текст совпадает с c.

Ниже будет доказано, что шифр сдвига является абсолютно стойким. Будем предполагать, что в качестве открытого текста шифротекста берется множество $\mathbb{Z}_n$, $n \in \mathbb{N}$ (см. стр. 35).

Теорема 1. *Предположим, что в шифре сдвига все возможные ключи являются равновероятными. Тогда для любого распределения вероятностей открытого текста шифр сдвига является абсолютно стойким.*

Доказательство. Пусть в шифре сдвига $M = C = K = \mathbb{Z}_n$, а

$$e(m,k) = (m+k) \bmod n,$$

$$d(c,k) = (c-k) \bmod n,$$

По условию теоремы $p_K(k) = 1/n$ для любого $k \in K$. Прежде всего, найдем закон распределения для множества C. Пусть $c \in \mathbb{Z}_n$, тогда

$$p_C(c) = \sum_{k \in \mathbb{Z}_n} p_K(k) p_M(d(c,k)) = \sum_{k \in \mathbb{Z}_n} \frac{1}{n} p_M\left((c-k) \bmod n\right) = \frac{1}{n}$$

поскольку при фиксированном $k \in \mathbb{Z}_n$ элементы $(c-k) \bmod n$, $k \in \mathbb{Z}_n$ пробегают все множество $\mathbb{Z}_n$ и выполняется равенство

$$\sum_{m \in \mathbb{Z}_n} p_M(m) = 1.$$

Для любых $m \in M$, $c \in C$ существует единственный ключ $k \in \mathbb{Z}_n$, такой, что $e(m,k) = c$ или $k = (m-c) \bmod n$. Отсюда получаем, что

$$p_C(c|m) = p_K((m-c) \bmod n) = \frac{1}{n}.$$

Теперь по формуле Байеса находим

$$p_M(m|c) = \frac{p_M(m)p_C(c|m)}{p_C(c)} = p_M(x),$$

поскольку $p_C(c|m) = p_C(c) = 1/n$, что и доказывает абсолютную стойкость. □

Итак, шифр сдвига нельзя «взломать», если для каждого символа открытого текста используется свой ключ, выбираемый случайным образом.

2.2. Криптоанализ

Современный криптоанализ опирается на такие математические науки как теория вероятностей и математическая статистика, алгебра, теория чисел, теория алгоритмов и ряд других. Все методы криптоанализа в целом укладываются в четыре направления в рамках которых, как правило, удается решить вопрос о криптостойкости алгоритма.

1. Статистический криптоанализ исследует возможности взлома криптосистем на основе изучения статистических закономерностей исходных и зашифрованных сообщений.

2. В рамках алгебраического криптоанализа ведется поиск математически слабых звеньев криптоалгоритмов. К примеру в 1997 году в эллиптических системах был выявлен класс ключей, которые существенно упрощали криптоанализ.

3. Дифференциальный (или разностный) криптоанализ основан на анализе зависимости изменения шифрованного текста от изменения исходного текста.

4. Линейный криптоанализ использует поиск линейной аппроксимации между исходным и шифрованным текстом.

Опыт взломов криптосистем показывает, что главным методом остается «лобовая» атака — проба на ключ. Также, как показывает опыт, криптосистемы страдают от небрежности в реализации.

2.3. Уязвимости организации процесса передачи сообщений

2.3.1. Человек посередине

Одна из самых популярных атак, которая заключается в том, что A и B устанавливают между собой «защищенный» канал связи. A и B используют криптографически надежные алгоритмы и ключи, однако, в результате их корреспонденция все равно становится доступной нарушителю E. Это нарушение приватности происходит из-за отсутствия

технологии проверки надежности используемых открытых ключей. Нарушитель E тем или иным способом внедряется в канал связи между A и B. Устанавливает с каждым собственную пару ключей, причем при передачи информации E может не только расшифровывать сообщение одним ключом и зашифровывать другим, но и видоизменять его так, чтобы оно выглядело более достоверно. Например, менять зашифрованный IP-адрес отправителя.

2.3.2. Обман, выполненный мафией

Рассмотрим следующую ситуацию. Пусть есть «Покупатель», который хочет купить шариковую ручку в «Магазине», но этот «Магазин» сотрудничает с «Клиентом». «Магазин» и «Клиент» действуют заодно и хотят обмануть «Покупателя» и «Банк». «Покупатель» и «Банк» действуют честно. «Покупатель» для оплаты шариковой ручки предъявляет пластиковую карточку или иной способ безналичной оплаты. «Магазин» обеспечивает «Клиента» возможностью представиться в «Банке» от имени «Клиента» и вывести значительную сумму со счета «Покупателя». В рамках такой схемы «Покупатель» и «Банк» работали честно, но мошенникам удалось вывести сумму со счета «Покупателя» в «Банке» за счет скоординированных действий «Магазина» и «Клиента».

Эти два рассмотренных случая демонстрируют важность не только криптостойкости алгоритма, но и надежности сценария использования этого алгоритма в реальной жизни с учетом поведения всех участников процесса.

2.3.3. Атака по сторонним (или побочным) каналам

Несмотря на надежность криптоалгоритма существует целый класс атак, позволяющий взламывать криптостойкие алгоритмы из-за уязвимостей в практической реализации алгоритма. Один из таких классов — атака по побочным каналам. В отличии от криптоанализа в данных атаках основное внимание уделяется особенностям реализации алгоритма, позволяющим узнать секретный ключ. Среди распространенных атак можно выделить следующие:

1. Атака по времени основана на предположении, что различные операции выполняются в устройстве за различное время, в зависимости от поданных входных данных. Таким образом, измеряя время вычислений и проводя статистический анализ данных, можно получить полную информацию о секретном ключе. Атака невозможна на алгоритмы, операции которых выполняются за одинаковое число тактов на всех платформах: вращение, сдвиг и другие битовые операции над фиксированным числом бит.

2. Атака по ошибкам вычислений заключается в осуществление различных воздействий на шифратор с целью возникновения искажения информации на некоторых этапах шифрования. Управляя этими искажениями и сравнивая результаты на разных этапах работы устройства, криптоаналитик может восстановить секретный ключ. Изучение атак на основе ошибок вычислений обычно разделяется на две ветви: одна изучает теоретические возможности для осуществления различных ошибок в исполнении алгоритма, другая исследует методы воздействия для реализации этих ошибок в конкретных устройствах.

3. Суть атаки по энергопотреблению состоит в том, что в процессе работы шифратора криптоаналитик с высокой точностью измеряет энергопотребление устройства и таким образом получает информацию о выполняемых в устройстве операциях и их параметрах.

4. Атака по электромагнитному излучению возможна против электронных устройств. Электронные шифрующие устройства излучают электромагнитное излучение во время своей работы. Связывая определённые спектральные компоненты этого излучения с операциями, выполняемыми в устройстве, можно получить достаточно информации для определения секретного ключа или самой обрабатываемой информации.

5. Акустическая атака появилась, наверное, раньше других и ассоциировалась с прослушкой работы принтеров и клавиатур, но в последние годы были найдены уязвимости, позволяющие использовать акустические атаки, направленные на внутренние компоненты электронных шифраторов.

Среди классических методов противодействию атакам по побочным каналам можно выделить следующие:

1. *Экранирование.* Достаточно сильное физическое экранирование устройства позволяет устранить почти все побочные каналы утечки информации. Недостатком экранирования является существенное увеличение стоимости и размеров устройства.

2. *Добавление шума.* Добавление шума существенно усложняет задачу криптоаналитика. Шумы уменьшают процент полезной информации в побочном канале, делая её нецелесообразной по затратам или вообще невозможной. Шум может быть добавлен как программно (введение случайных вычислений), так и аппаратно (установка различных генераторов шума).

3. *Уравнивание времени выполнения операций.* Чтобы криптоаналитик не смог провести атаку по времени исполнения все этапы шифрования в устройстве должны выполняться за одинаковое время.

4. *Балансировка энергопотребления.* По возможности при проведении операций должны быть задействованы все аппаратные части устройства (регистры или вентили), на неиспользуемых частях следует проводить ложные вычисления. Таким образом, можно добиться постоянства энергопотребления устройством и защититься от атак по энергопотреблению.

5. *Устранение условных переходов.* Защититься от множества атак по сторонним каналам можно, устранив в алгоритме операции условного перехода, зависящие от входных данных или секретного ключа. В идеале алгоритм вообще не должен содержать операторов ветвления, и все вычисления должны производиться с помощью элементарных побитовых операций.

2.4. Парадокс дней рождения

Рассмотрим студенческую группу из n человек, и пусть дни рождения в этой группе распределены равномерно, то есть нет високосных дат, близнецов, рождаемость не зависит от дня недели, времени года и других факторов. Какова будет вероятность $p(n)$ того, что у двух человек из группы дни рождения совпадут? Рассчитаем сначала вероятность дополнительного события $\bar{p}(n)$, что в группе из n человек дни рождения всех людей будут различными. Если $n > 365$, то в силу принципа Дирихле вероятность равна нулю. Если же $n \leqslant 365$, то возьмём наугад одного человека из группы и запомним его день рождения. Затем возьмём наугад второго человека, при этом вероятность того, что у него день рождения не совпадёт с днем рождения первого человека, равна $1 - \frac{1}{365}$. Затем возьмём третьего человека — при этом вероятность того, что его день рождения не совпадёт с днями рождения первых двух, равна $1 - \frac{2}{365}$. Рассуждая по аналогии, мы дойдём до последнего человека, для которого вероятность несовпадения его дня рождения со всеми предыдущими будет равна $1 - \frac{n-1}{365}$. Перемножая все эти вероятности, получаем вероятность того, что все дни рождения в группе будут различными:

$$\bar{p}(n) = 1 \cdot \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{365}\right). \qquad (2.1)$$

Оценим значение $\bar{p}(n)$. Поскольку $e^x \approx 1 + x$ при достаточно малом x, то множители входящие в (2.1) можно оценить как

$$\left(1 - \frac{i}{365}\right) \approx e^{-\frac{i}{365}}.$$

Произведение множителей оценивается как $\bar{p}(n) \approx e^{-\frac{(1+2+\ldots+(n-1))}{365}} = e^{-\frac{n(n-1)}{2 \cdot 365}} \approx e^{-\frac{n^2}{2 \cdot 365}}$, тогда вероятность того, что хотя бы у двух человек из

n дни рождения совпадут, равна

$$p(n) = 1 - \bar{p}(n) \approx 1 - e^{-\frac{n^2}{2 \cdot 365}}.$$

Вероятности для некоторых значений n иллюстрируются таблицей 2.1.

Таблица 2.1
Парадокс дней рождения

n	$p(n)$
10	12%
20	41%
23	51%
30	70%
50	97%
367	100%

В криптографии на этом парадоксе основана атака на электронную цифровую подпись, например. Допустим, что А и Б — хотят подписать контракт, но А хочет подложить Б поддельный вариант контракта. Тогда А составляет подлинный контракт и поддельный контракт. Далее посредством внесения допустимых изменений, не меняющих смысла текста (расстановкой запятых, переносов, отступов), А добивается, чтобы оба контракта имели одинаковый хэш (см. стр. 128). После этого А посылает Б подлинный контракт, Б его подписывает, а его подпись также показывает, что он подписал и поддельный контракт, так как оба контракта имеют одинаковый хэш. Для избежания уязвимости такого рода достаточно увеличить длину хэша настолько, чтобы стало вычислительно сложно найти два контракта с одинаковыми хэшами. В частности, требуется в два раза большая длина хэша, чтобы обеспечить вычислительную сложность, сравнимую со сложностью полного перебора.

2.5. Полный перебор

Метод полного перебора — метод решения математических задач. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий. Любая задача из класса NP может быть решена полным перебором. В криптографии на вычислительной сложности полного перебора основывается оценка криптостойкости шифров. В частности, шифр считается криптостойким, если не существует метода «взлома» существенно более быстрого, чем полный перебор всех ключей. Криптографические атаки, основанные на методе полного перебора, являются самыми универсальными, но и самыми долгими.

Можно предположить, что с ростом мощности компьютеров разрядность ключа, достаточная для обеспечения безопасности информации против атаки методом полного перебора, будет неограниченно расти. Однако это не так. Существуют фундаментальные ограничения вычислительной мощности, наложенные структурой Вселенной: например, скорость передачи любого сигнала не может превышать скорость распро-

странения света в вакууме, а количество атомов во Вселенной (из которых, в конечном счете, состоят компьютеры) огромно, но конечно. Так, например, можно описать два фундаментальных ограничения.

Любому вычислительному устройству требуется энергия для проведения вычислений. В нашей солнечной системе основным источником энергии является Солнце. Мощность излучения Солнца составляет примерно $3.8 \cdot 10^{26}$ Вт. Возраст Солнца на текущее время составляет около $4.5 \cdot 10^9$ лет, ожидается, что Солнце всего просуществует 10–12 млрд. лет. Для проведения одной операции требуется энергия порядка kT, где $k = 1.4 \cdot 10^{-23}$ — постоянная Больцмана, а T — абсолютная температура вычислителя. Таким образом, за все время существования вычислителя удастся провести около

$$\frac{3.8 \cdot 10^{26} \cdot 10^{10} \cdot 365 \cdot 24 \cdot 60 \cdot 60}{1.4 \cdot 10^{-23} \cdot 10^{-2}} \approx 10^{70} \approx 2^{235}$$

операций или подобрать 235 бит ключа.

Любое вычислительное устройство имеет какие-либо физические размеры. Предположим, что удалось создать вычислитель размером с протон, т.е. массой $1.6 \cdot 10^{-27}$кг. Масса Земли — $6 \cdot 10^{24}$ кг, т.е. если превратить всю Землю в вычислители, то можно построить $6 \cdot 10^{24} / 1.6 \cdot 10^{-27} \approx 4 \cdot 10^{51}$ вычислителей. Предположим, что время, необходимое такому вычислителю для проведения одной операции, соразмерно времени, которое затрачивает луч света для преодоления расстояния равного диаметру протона, т. е. одна операция занимает $0.8 \cdot 10^{-15} / 3 \cdot 10^8 \approx 3 \cdot 10^{-24}$ с. За все предполагаемое время существования солнечной системы 10^{10} лет, эти компьютеры, работая параллельно, смогут выполнить

$$\frac{4 \cdot 10^{51} \cdot 10^{10} \cdot 365 \cdot 24 \cdot 60 \cdot 60}{3 \cdot 10^{-24}} \approx 10^{92} \approx 2^{308}$$

операций или подобрать 308 бит ключа.

Минимальный размер ключа, необходимый для защиты информации от атак злоумышленника, будет расти. По мере повышения быстродействия компьютеров, тем не менее приведенные вычисления показывают, что существует возможность выбрать такую длину ключа, что атаку методом полного перебора будет провести в принципе невозможно, вне зависимости от повышения вычислительной мощности компьютеров или успехов в области классической теории алгоритмов.

Упражнение 1. Сколько бит ключа может вскрыть самый мощный компьютер из списка TOP500 за 100 лет работы (предполагайте, что одна операция позволяет проверить один вариант ключа)?

3. СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

Симметричные алгоритмы шифрования (или криптография с секретными ключами) основаны на том, что отправитель и получатель информации используют один и тот же ключ. Этот ключ должен храниться в тайне и передаваться способом, исключающим его перехват (рис. 3.1).

Обмен информацией осуществляется в три этапа:

1. Отправитель передает получателю ключ (в случае сети с несколькими абонентами у каждой пары абонентов должен быть свой ключ, отличный от ключей других пар).

2. Отправитель, используя ключ, зашифровывает сообщение, которое пересылается получателю.

3. Получатель получает сообщение и расшифровывает его.

Если для каждого сеанса связи будет использоваться уникальный ключ, это повысит защищенность системы. Криптографические алгоритмы обычно строятся с использованием простых и быстро выполняемых операторов нескольких типов. Среди всевозможных классов симметричных шифров выделяют:

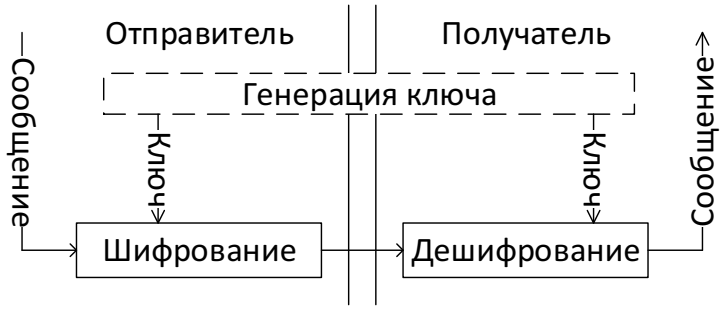


Рис. 3.1. Схема симметричного шифрования

1. Блочные шифры — это шифры, которые выполняют шифрование небольшими блоками бит. Их можно рассматривать как подстановку на множестве блоков. Например, шифры ГОСТ 28147-89, AES, DES.

2. Поточные шифры — это шифры, в которых каждый символ исходного текста преобразуется в символ шифротекста в зависимости не только от самого символа, но и его места в исходном тексте. Например, шифры RC4, A5. Существуют способы, позволяющие с помощью блочного шифра организовать поточный режим шифрования.

3. Шифры гаммирования — это шифры, основанные на наложении (например, сложение по модулю 2) определенной последовательности, называемой гаммой, на открытый текст. Как правило, длина гаммы выбирается очень большой.

Качество алгоритма шифрования очень сильно зависит от возникающего лавинного эффекта в в процессе шифрования. Говорят, что криптографический алгоритм удовлетворяет лавинному критерию, если при

изменении одного бита входной последовательности изменяется в среднем половина выходных битов. Лавинный эффект является следствием хорошей конфузии и диффузии.

Диффузия — метод, при котором избыточность в статистике входных данных «распределяется» по всей структуре выходных данных. При этом для статистического анализа требуются большие объёмы выходных данных, скрывается структура исходного текста. Реализуется при помощи P-блоков.

Конфузия — метод, при котором зависимость ключа и выходных данных делается как можно более сложной, в частности, нелинейной. При этом становится сложнее делать заключения о ключе по выходным данным, а также об исходных данных, если известна часть ключа. Реализуется при помощи S-блоков.

Все многообразие существующих операций в криптографических методах с секретным ключом можно свести к двум классам преобразований: перестановки (P-блокам) и подстановки (S-блокам).

3.1. Подстановочно-перестановочная сеть

Существует несколько классических схем построения шифров. Мы рассмотрим две из них: сеть Фейстеля и подстановочно-перестановочную сеть. В качестве стандартных частей шифра в этих схемах используются S-блоки и P-блоки.

P-блок — блок перестановок (Permutation). Работа P-блока заключается в перестановке бит входного слова. Принципиальная схема работы блока приведена на рис. 3.2. В результате работы P-блока в восьмибитном блоке единичные биты были переставлены с третьей на нулевую позицию и с пятой на седьмую.

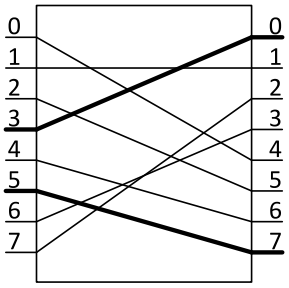


Рис. 3.2. Принципиальная схема работы P-блока

S-блок — блок подстановок (Substitution). Принципиальная схема работы блока приведена на рис. 3.3. Работа любого S-блока заключается в подстановке вместо входного слова некоторого выходного и может быть записана в виде таблицы 3.1. Для сокращенной записи можно выписывать только последовательность выходных значений, предполагая, что входные значения упорядочены по возрастанию. В таблице 3.1 S-блок задан как (6, 2, 7, 5, 0, 3, 1, 4).

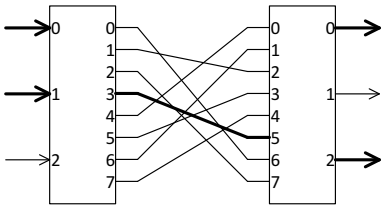


Рис. 3.3. Принципиальная схема работы S-блока

Подстановочно-перестановочная сеть (SP-сеть) — это разновидность блочного шифра, который, в общем случае, состоит из чередующихся слоев S-блоков и P-блоков. Пример такого шифра представлен на рис. 3.4.

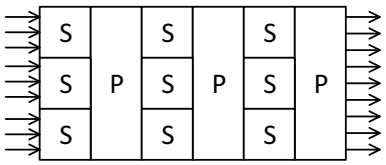


Рис. 3.4. Принципиальная схема SP-сети

Конкретный алгоритм определяет каким образом в SP-сети будет реализована зависимость выходного вектора от ключа. Например, алгоритм в P-блоке может выполнять сложение текущего значения и ключа или, в зависимости от бит ключа, будут использованы разные S-блоки.

Таблица 3.1

Подстановка S-блока

Входное значение	0 000_2	1 001_2	2 010_2	3 011_2	4 100_2	5 101_2	6 110_2	7 111_2
Выходное значение	6 110_2	2 010_2	7 111_2	5 101_2	0 000_2	3 011_2	1 001_2	4 100_2

Упражнение 2. Каково количество различных S-блоков для двоичных слов длины n?

Упражнение 3. Каково количество различных P-блоков для двоичных слов длины n?

3.2. Сеть Фейстеля

Сеть Фейстеля для блочного шифра представляет собой повторение однотипных раундов (рис. 3.5). В каждом раунде исходный блок разделяется на две равные половины L_i и R_i, где i — номер раунда. После чего правая половина без изменений записывается на левую половину выходного вектора и подается на вход в некоторую нелинейную функцию $F(R_i, K_i)$. Функция $F(R_i, K_i)$ принимает правую половину и раундовый ключ K_i алгоритма. Результат вычисления функции $F(R_i, K_i)$ складывается по модулю 2 с левой половиной входного блока и записывается на правую половину выходного вектора. Подобные раунды повторяются

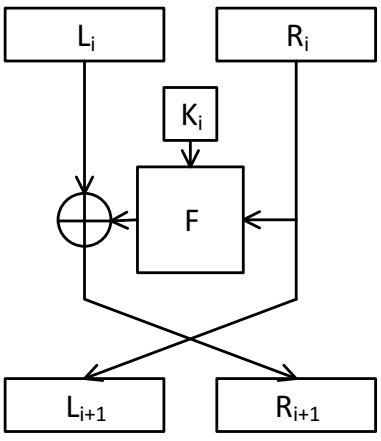


Рис. 3.5. Один раунд сети Фейстеля

многократно. Особенностью сети Фейстеля является то, что расшифровывание выполняется так же, как и зашифровывание, но инвертируется порядок подключей K_i. Для задания алгоритма требуется указать как формируются раундовые ключи и точный вид нелинейной функции $F(R_i, K_i)$, общее количество выполняемых раундов, размер блока.

Упражнение 4. Покажите, что шифрование (порядок подключей $K_1...K_n$) и дешифрование (порядок подключей $K_n...K_1$) взаимнообратные функции.

3.3. Стандарт шифрования ГОСТ 28147-89

Основой алгоритма ГОСТ 28147-89 является сеть Фейстеля. Всего в сети Фейстеля выполняется 32 раунда. Размер блока — 64 бита. Соответственно, L_i и R_i имеют размер по 32 бита. Ключ алгоритма имеет длину 256 бит. Раундовые ключи K_i генерируются следующим образом. Исходный 256-битный ключ разбивается на восемь 32-битных блоков — это будут первые восемь раундовых ключей $K_1...K_8$. Ключи $K_9...K_{16}$, $K_{17}...K_{24}$ являются циклическим повторением ключей $K_1...K_8$. Ключи $K_{25}...K_{32}$ являются ключами $K_8...K_1$.

Функция$F(R_i, K_i)$ вычисляется следующим образом: R_i и K_i складываются по модулю 2^{32}. Результат разбивается на восемь 4-битовых подпоследовательностей, каждая из которых поступает на вход своего узла таблицы замен (в порядке возрастания старшинства битов), называемого ниже S-блоком. Общее количество S-блоков ГОСТа — восемь, то есть, столько же, сколько и подпоследовательностей. Каждый S-блок представляет собой перестановку чисел от 0 до 15 (конкретный вид S-блоков в стандарте не определен). Первая 4-битная подпоследовательность попадает на вход первого S-блока, вторая — на вход второго и т. д.

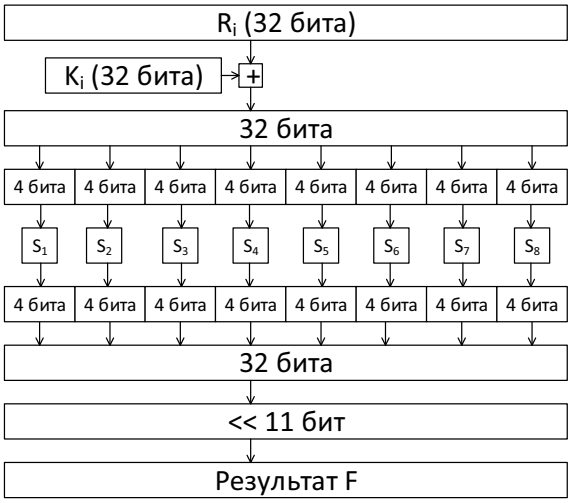


Рис. 3.6. Нелинейная функция F сети Фейстеля для алгоритма ГОСТ 28147-89

Все восемь S-блоков могут быть различными. В ГОСТ 28147-89 конкретные S-блоки не заданы, но чаще всего используют S-блоки ЦБ РФ.

Алгоритм ГОСТ 28147-89 предусматривает следующие режимы использования:

Таблица 3.2

S-блоки Центрального банка РФ

Номер S-блока	Значение
1	4 10 9 2 13 8 0 14 6 11 1 12 7 15 5 3
2	14 11 4 12 6 13 15 10 2 3 8 1 0 7 5 9
3	5 8 1 13 10 3 4 2 14 15 12 7 6 0 9 11
4	7 13 10 1 0 8 9 15 14 4 6 12 11 2 5 3
5	6 12 7 1 5 15 13 8 4 10 9 14 0 3 11 2
6	4 11 10 0 7 2 1 13 3 6 8 5 9 12 15 14
7	13 11 4 1 3 15 5 9 0 10 14 7 6 8 2 12
8	1 15 13 0 5 7 10 4 9 2 3 14 6 11 8 12

1. **Режим простой замены** — один из вариантов использования симметричного блочного шифра, при котором каждый блок открытого текста заменяется блоком шифротекста. Шифрование может быть описано следующим образом: $C_i = E_k(M_i)$, где i — номера блоков,C_i и M_i — блоки зашифрованного и открытого текстов соответственно, а E_k — функция блочного шифрования. Расшифровка аналогична: $M_i = D_k(C_i)$. Среди преимуществ можно выделить возможность параллельного шифрования, устойчивость к появлению ошибок передачи — ошибки не распространяются на другие блоки. Недостатком является плохое сокрытие структуры сообщения — блоки могут повторяться.

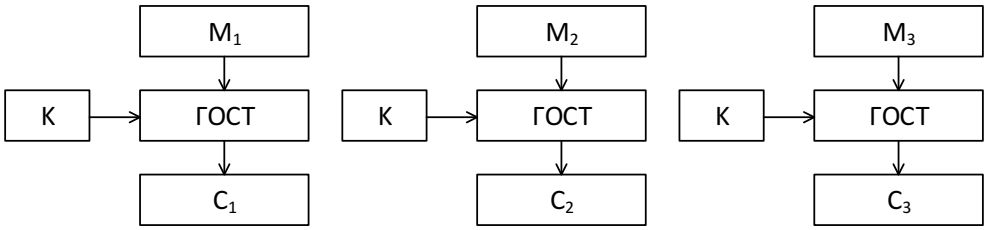


Рис. 3.7. Режим простой замены

2. **Режим гаммирования с обратной связью** — один из вариантов использования симметричного блочного шифра, при котором для шифрования следующего блока открытого текста он складывается по модулю 2 с перешифрованным (блочным шифром) результатом шифрования предыдущего блока. Шифрование может быть описано следующим образом:$C_0 = IV\, C_i = E_k(C_{i-1}) \oplus M_i\, M_i = E_k(C_{i-1}) \oplus C_i$, где i — номера блоков, IV — вектор инициализации (синхропосылка), C_i и M_i — блоки зашифрованного и открытого текстов соответственно, а E_k — функция блочного шифрования. Ошибка, которая возникает в шифротексте при передаче (например, из-за помех), сделает невозможным расшифровку

как блока, в котором ошибка произошла, так и следующего за ним, однако не распространяется на последующие блоки.

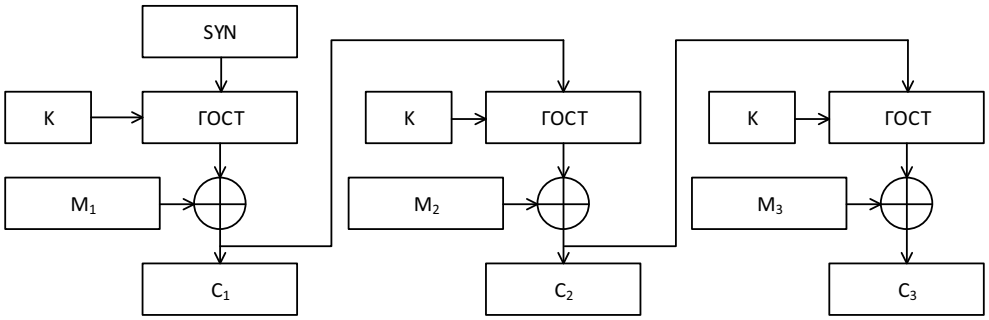


Рис. 3.8. Режим гаммирования с обратной связью

3. **Режим гаммирования** — один из режимов шифрования для симметричного блочного шифра с использованием механизма обратной связи. Каждый блок открытого текста (кроме первого) побитово складывается по модулю 2 (операция XOR) с предыдущим результатом шифрования. Шифрование может быть описано следующим образом: $C_0 = IV\, C_i = E_k\left(M_i \oplus C_{i-1}\right)$, где i — номера блоков, IV — вектор инициализации (синхропосылка), C_i и M_i — блоки зашифрованного и открытого текстов соответственно, а E_k — функция блочного шифрования. Расшифровка: $M_i = C_{i-1} \oplus D_k\left(C_i\right)$. Если при передаче произойдёт изменение одного бита шифротекста, данная ошибка распространится и на следующий блок. Однако на последующие блоки (через один) ошибка не распространится. Для очень крупных сообщений (32 Гбайта при длине блока 64 бита) всё-таки возможно применение атак, основанных на структурных особенностях открытого текста (следствие парадокса дней рождения).

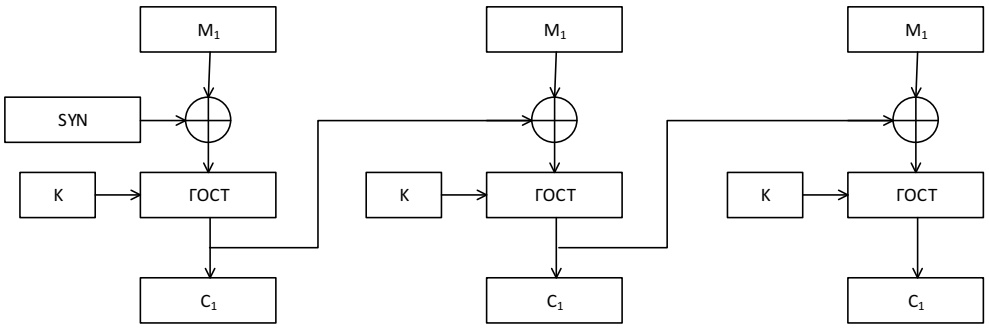


Рис. 3.9. Режим гаммирования

Упражнение 5. Проанализируйте криптостойкость алгоритма ГОСТ 28147-89, если в качестве S-блоков выбрать блоки вида 0, 1, 2, ...14, 15.

4. ВВЕДЕНИЕ В ТЕОРИЮ ЧИСЕЛ

В арифметике целых чисел используется множество целых чисел, обозначаемых $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$ и несколько операций. В криптографии нас интересуют три бинарных операции в приложении к множеству целых чисел. Бинарные операции имеют два входа и один выход

$$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}.$$

Деление не относится к таким операциям, потому что при делении целых чисел возникает два выхода вместо одного.

Будем говорить, что делим число a на число n с остатком, когда представляем a в виде $a = q \cdot n + r$, где $a \in \mathbb{Z}$ называется делимое, $n \in \mathbb{N}$ — делитель, $q \in \mathbb{Z}$ — неполное частное, а $r \in \mathbb{Z}_+$, $0 \leq r < n$ — остаток.

Теорема 2. *Для любого целого числа a и любого натурального n существует и единственна пара целых чисел q и r таких, что $a = qn + r$ и $0 \leq r < n$.*

Доказательство. Рассмотрим числа $..., -3n, -2n, -n, 0, n, 2n, 3n, ...$ Число a будет находиться между какими-то соседними из этих чисел. Предположим, что между числами $qn \leq a < (q+1)n$. Вычтем из всех частей двойного неравенства qn, получим $0 \leq a - qn < n$. Обозначим $r = a - qn$ и получим искомое представление числа $a = qn + r$.

Докажем единственность от противного. Пусть существуют две пары (q_1, r_2) и (q_2, r_2), такие что $q_1 \neq q_2$ и $r_1 \neq r_2$. Тогда $a = q_1 n + r_1 = q_2 n + r_2$. Следовательно, $n(q_1 - q_2) = r_2 - r_1$. А поскольку, $|n(q_1 - q_2)| \geq n$ и $|r_1 - r_2| < n$, то получено противоречие. □

Например, поделим -255 на 11, тогда потребуется решить уравнение $-255 = q \cdot 11 + r$, $q \in \mathbb{Z}$, $0 \leq r < n$. Поделим столбиком $-255/11 = -23, (18)$. Округлим полученное значение вниз, получим $q = -24$. Найдем r по формуле $r = a - q \cdot n$, $r = -255 - (-24) \cdot 11 = -255 + 264 = 9$. Ответ можно записать как $-255 = (-24) \cdot 11 + 9$.

Если a не равно нулю, а $r = 0$, то мы имеем $a = q \cdot n$. Введем следующие обозначения: $a \vdots b$ означает, что a делится на b, или число a кратно числу b. Запись $b \mid a$ означает, что b делит a, или, что то же самое: b — делитель a. Если остаток при делении не является нулевым, то a не делится нацело, и мы можем записать это как $b \nmid a$.

Положительное целое число может иметь больше чем один делитель. Например, целое число 32 имеет шесть делителей: 1, 2, 4, 8, 16 и 32. Любое натуральное число, кроме единицы, имеет по крайней мере два делителя — 1 и само себя.

Определение 5. Натуральное число p называется *простым*, если оно имеет ровно два делителя: единицу и само себя.

Определение 6. Натуральное число n называется *составным*, если оно имеет более двух делителей.

Натуральные числа можно разделить на составные числа (более двух делителей), простые числа (ровно два делителя) и на число 1 (точно один делитель).

Определение 7. Два числа называются *взаимно простыми*, если они не имеют общих делителей, кроме ± 1.

Число 1 является взаимно простым с любым числом, простое число p взаимно просто со всеми числами от 1 до $p-1$.

Упражнение 6. Докажите, что если $a \mid b$ и $b \mid a$, то $a = \pm b$.

Упражнение 7. Докажите, что если $a \mid b$ и $b \mid c$, то $a \mid c$.

Упражнение 8. Докажите, что если $a \mid b$ и $a \mid c$, то $a \mid (m \cdot b + n \cdot c)$, где m и n — произвольные целые числа.

4.1. Модульная арифметика

Рассмотрим остаток при деления на n числа a, обозначаемые следующим образом

$$a \bmod n = r.$$

Так, например, $27 \bmod 5 = 2$, $36 \bmod 12 = 0$, $-18 \bmod 14 = 10$ и $17 \bmod 5 = 2$. Обратим внимание, что остатки от деления и 27 и 17 на 5 совпали и были равны 2.

Определение 8. Два целых числа *сравнимы по модулю* n, если при делении на n они дают одинаковые остатки.

Из этого определения следует, что если два числа a, b сравнимы по модулю n, то

$$a \bmod n = b \bmod n.$$

Записывая остатки для чисел a и b при делении на n, получаем

$$\begin{cases} a = q_1 n + r; \\ b = q_2 n + r. \end{cases}$$

Следовательно, два числа a, b сравнимы по модулю n тогда и только тогда, когда $a - b = q_1 n + r - q_2 n - r = n(q_1 - q_2) \vdots n$. Утверждение, что числа a, b сравнимы по модулю n, обозначается с помощью следующей формальной записи $a \equiv b \pmod{n}$, называемой отношением сравнения.

Отношение сравнения является отношением эквивалентности. Все числа, сравнимые по модулю n, образуют класс вычетов по модулю n. Любое число в классе называется вычетом по модулю n. Всем числам класса вычетов соответствует один и тот же остаток r, а поскольку всего имеется n остатков: 0, 1, , ..., $n-1$, то существует n различных классов вычетов. Обычно класс вычетов обозначается наименьшим неотрицательным элементом класса вычетов. Выбрав из каждого класса вычетов по модулю n по одному вычету, получим полную систему вычетов по модулю n. Выберем в качестве $n = 3$, тогда у нас индуцируются три класса вычетов $[0] = \{..., -9, -6, -3, 0, 3, 6, 9, ...\}$, $[1] = \{..., -8, -5, -2, 1, 4, 7, 10, ...\}$, $[2] = \{..., -7, -4, -1, 2, 5, 8, 11, ...\}$. Далее, для упрощения записи, квадратные скобки мы будем опускать. Множество всех классов вычетов по модулю n обозначается как $\mathbb{Z}_n$. Над элементами множества $\mathbb{Z}_n$ можно выполнять операции сложения, вычитания, умножения как над целыми числами. Рассмотрим следующий пример.

Пример 1. Выполните сложение 7 и 14 в $\mathbb{Z}_{15}$. Множество $\mathbb{Z}_{15}$ имеет вид $\{[0], [1], ...[14]\}$, а элементы 7 и 14 задают классы вычетов $7+15k_1$, $k_1 \in \mathbb{Z}$ и $14+15k_2$, $k_2 \in \mathbb{Z}$ соответственно. Сложим их, получим класс вычетов $7+15k_1+14+15k_2 = 21+15(k_1+k_2) = 21+15k_3$, $k_3 \in \mathbb{Z}$, где $k_1+k_2 = k_3$, т. к. области значения левой и правой части совпадают. Заменим k_3 на $k-1$ и получим $21+15k_3 = 21+15(k-1) = 21-15+15k = 6+15k$. Следовательно, получили класс вычетов [6]. Этот же ответ можно получить быстрее по формуле $(7+14) \bmod 15 = 6$.

Пример 2. Выполните вычитание 11 из 7 в $\mathbb{Z}_{13}$. Множество $\mathbb{Z}_{13}$ имеет вид $\mathbb{Z}_{13} = \{[0], [1], ...[12]\}$, а элементы 11 и 7 задают классы вычетов $11+13k_1$, $k_1 \in \mathbb{Z}$ и $7+13k_2$, $k_2 \in \mathbb{Z}$ соответственно. Вычтем их, получим класс вычетов $11+13k_1-7-13k_2 = 4+13(k_1-k_2) = 4+13k$, $k \in \mathbb{Z}$, где $k_1-k_2 = k_3$, т. к. области значения левой и правой части совпадают. Следовательно, получили класс вычетов [4]. Этот же ответ можно получить быстрее по формуле $(11-7) \bmod 13 = 4$.

Пример 3. Выполните умножение 11 на 7 в $\mathbb{Z}_{20}$. Множество $\mathbb{Z}_{20}$ имеет вид $\mathbb{Z}_{20} = \{[0], [1], ...[19]\}$, а элементы 11 и 7 задают классы вычетов $11+20k_1$, $k_1 \in \mathbb{Z}$ и $7+20k_2$, $k_2 \in \mathbb{Z}$ соответственно. Перемножим их, получим класс вычетов $(11+20k_1)(7+20k_2) = 77+140k_1+220k_2+400k_1k_2 = 77+20(7k_1+11k_2+20k_1k_2) = 77+20k_3$, $k_3 \in \mathbb{Z}$, где $7k_1+11k_2+20k_1k_2 = k_3$, т. к. области значения левой и правой части совпадают. Заменим k_3 на $k-3$ и получим $77+20k_3 = 77+20(k-3) = 77-60+20k = 17+20k$. Следовательно, получили класс вычетов [17]. Этот же ответ можно получить быстрее по формуле $(11 \cdot 7) \bmod 20 = 17$.

Рассмотрим некоторые свойства сравнений.

Предложение 1. *Если a, b, c и n — целые числа, $n > 0$, такие, что $a \equiv b \pmod{n}$, то*

1. $a + c \equiv b + c \pmod{n}$,

2. $a - c \equiv b - c \pmod{n}$,

3. $a \cdot c \equiv b \cdot c \pmod{n}$.

Доказательство. Обратим внимание, что $a \equiv b \pmod{n}$ означает, что $n|(a - b)$.

1.$(a + c) - (b + c) = a - b$. Поскольку $n|(a - b)$, $n|(a + c) - (b + c)$. Поэтому $a + c \equiv b + c \pmod{n}$.

2. $(a - c) - (b - c) = a - b$. Поскольку $n|(a - b)$, $n|(a - c) - (b - c)$. Поэтому $a - c \equiv b - c \pmod{n}$.

3.$(a \cdot c) - (b \cdot c) = (a - b) \cdot c$. Поскольку $n|(a - b)$, $n|(a - b) \cdot c$. Поэтому $a \cdot c \equiv b \cdot c \pmod{n}$. □

Предложение 2. *Если a, b, c, d и n — целые числа, $n > 0$ такие, что $a \equiv b \pmod{n}$ и $c \equiv d \pmod{n}$, то*

1. $a + c \equiv b + d \pmod{n}$,

2. $a - c \equiv b - d \pmod{n}$,

3. $a \cdot c \equiv b \cdot d \pmod{n}$.

Доказательство. Обратим внимание, что $a \equiv b \pmod{n}$ означает $n|(a - b) = k \cdot n$, $k \in \mathbb{Z}$; $c \equiv d \pmod{n}$ означает $n|(c - d) = f \cdot n$, $f \in \mathbb{Z}$

1. $(a + c) - (b + d) = (a - b) + (c - d) = k \cdot n + f \cdot n = (k + f) \cdot n$. Поэтому$(a + c) \equiv (b + d) \pmod{n}$.

2. $(a - c) - (b - d) = (a - b) - (c - d) = k \cdot n - f \cdot n = (k - f) \cdot n$. Поэтому $a - c \equiv b - d \pmod{n}$.

3. $a \cdot c - b \cdot d = c \cdot (a - b) + b \cdot (c - d) = (c \cdot k + b \cdot f) \cdot n$. Поэтому $a \cdot c \equiv b \cdot d \pmod{n}$. □

Пример 4. Докажите, что остаток от целого числа, разделенного на 3, такой же, как остаток от суммы деления его десятичных цифр.

Пусть число a имеет следующую десятичную запись $a = \sum\limits_{k=0}^{n} a_k 10^k$. Тогда

$$a \bmod 3 = \left(\sum_{k=0}^{n} a_k 10^k\right) \bmod 3 = \sum_{k=0}^{n} \left[\left(a_k 10^k\right) \bmod 3\right] =$$

$$= \sum_{k=0}^{n} \left(a_k \bmod 3\right) \left(10^k \bmod 3\right) = \sum_{k=0}^{n} \left(a_k \bmod 3\right) \left(10 \bmod 3\right)^k =$$

$$= \sum_{k=0}^{n} (a_k \bmod 3)\, 1^k = \left(\sum_{k=0}^{n} a_k \right) \bmod 3.$$

Упражнение 9. Докажите, что отношение сравнения является отношением эквивалентности.

Упражнение 10. Докажите, что если $a \equiv b \pmod{m_1}$ и $a \equiv b \pmod{m_2}$, то $a \equiv b \pmod{m}$, где $m = \text{НОК}(m_1, m_2)$.

Упражнение 11. Докажите, что если $a \equiv b \pmod{m}$, то a, b сравнимы по любому модулю — делителю m.

4.2. Наибольший общий делитель

Каждое число может иметь несколько делителей. Рассмотрим числа 12 и 140, у них есть делители есть 1, 2 и 4. Однако наибольший общий делитель — 4 (рис. 4.1). Два положительных целых числа могут иметь несколько общих делителей, но только один наибольший общий делитель.

Таблица 4.1

Общие делители двух целых чисел

Делители	1	2	3	4	5	6	7	10	12	14	20	28	35	70	140
12	+	+	+	+		+			+						
140	+	+		+	+		+	+		+	+	+	+	+	+

Определение 9. *Наибольшим общим делителем* двух положительных целых чисел называется такое наибольшее целое число, которое делит оба целых числа.

Предложение 3. *Если a, b, c и n — целые числа, $n > 0$, $c \neq 0$, такие, что $ac \equiv bc \pmod{n}$ и НОД$(c, n) = 1$, то $a \equiv b \pmod{n}$.*

Доказательство. Обратим внимание, что $ac \equiv bc \pmod{n}$ означает, что $n|(ac - bc)$. Следовательно, $n|(a - b)c$, но поскольку НОД$(c, n) = 1$, то $n|(a - b)$. □

Предложение 4. *Если a, b, c и n — целые числа, $n > 0$, $c \neq 0$, такие, что $ac \equiv bc \pmod{nc}$, то $a \equiv b \pmod{n}$.*

Доказательство. Обратим внимание, что $ac \equiv bc \pmod{nc}$ означает, что $nc|(ac - bc)$. Следовательно, $(a - b)c = qnc$. Поделим левую и правую часть на c. Получим $a - b = qn$ или $n|(a - b)$. □

Нахождение наибольшего общего делителя (НОД) двух положительных целых чисел путем составления списка всех общих делителей непригодно для достаточно больших чисел. Для поиска НОД существует эффективный алгоритм — алгоритм Евклида.

4.2.1. Алгоритм Евклида

Евклид (300 г. до н. э.) разработал алгоритм, который может найти наибольший общий делитель двух положительных целых чисел. Алгоритм Евклида основан на следующих двух фактах:

Предложение 5. *НОД*$(a, 0) = a$.

Доказательство. Из определения наибольшего общего делителя следует достоверность предложения. □

Предложение 6. *НОД*$(a, b) =$ *НОД*(b, r), *где* r — *остаток от деления* a *на* b.

Доказательство. Если r — остаток от деления a на b, то $a = b \cdot q + r$, где $q \in \mathbb{Z}$. Пусть E — множество всех общих делителей a и b. Каждый элемент E делит a и b, поэтому он делит $a - b \cdot q = r$. Это означает, что E — множество всех общих делителей a, b, и r. Предположим, что F — множество всех общих делителей b и r. Каждый элемент F делит b и r, поэтому делит $b \cdot q + r = a$. Т. е. F — множество всех общих делителей a, b и r. Это означает, что $E = F$ и, следовательно, НОД$(a, b) =$ НОД(b, r). □

Например, вычисляя НОД (36, 10) с помощью алгоритма 4.1, мы можем использовать второй факт несколько раз и один раз первый факт следующим образом: НОД$(36, 10) =$ НОД$(10, 6) =$ НОД$(6, 4) =$ НОД$(4, 2) =$ НОД$(2, 0) = 2$.

Алгоритм 4.1 Алгоритм Евклида

```
1: procedure EUCLID(a, b)
2:     if b = 0 then
3:         return a
4:     else
5:         return Euclid(b, a mod b)
6:     end if
7: end procedure
```

Пример 5. Найдем наибольший общий делитель 2740 и 1760.

Применим вышеупомянутую процедуру, используя таблицу 4.2. В таблице также показаны значения q на каждом шаге. Мы имеем НОД$(2740, 1760) = 20$. Заполнение таблицы происходит сверху вниз.

Таблица 4.2
Пример вычисления НОД$(2740, 1760)$

Номер строки	a	b	q	r
1	2740	1760	1	980
2	1760	980	1	780
3	980	780	1	200
4	780	200	3	180
5	200	180	1	20
6	180	20	9	0
7	**20**	0	—	—

Упражнение 12. Найти наибольший общий делитель 25 и 60.

4.2.2. Расширенный алгоритм Евклида

Пусть даны два целых числа a и b. Нам надо найти другие два целых числа s и t, такие что

$$s \cdot a + t \cdot b = \text{НОД}(a, b).$$

Расширенный алгоритм Евклида может вычислить НОД(a, b) и в то же самое время вычислить значения s и t.

Теорема 3. *Если a и b — целые числа и оба не равны нулю, то существуют целые числа s и t такие, что НОД$(a, b) = s \cdot a + t \cdot b$.*

Доказательство. Предположим, что D — множество всех значений вида $s \cdot a + t \cdot b$, а d — есть наименьшее значение, отличное от нуля. Мы можем записать $a = q \cdot d + r$ или $r = a - q \cdot d = (1 - q \cdot x) \cdot a + (-q \cdot y) \cdot b$, где $0 \leq r < d$. Это означает, что r входит в D. Но поскольку $r < d$, то $r = 0$ и, следовательно, $d|a$. Аналогично $d|b$. Поэтому d — общий делитель a и b. Любой другой делитель a и b делит и $d = x \cdot a + y \cdot b$ тоже. Поэтому d должен быть равен НОД(a, b). □

Предложение 7. *Если для целых чисел a и b число $d =$ НОД(a, b) и числа s' и t' удовлетворяют соотношению $d = b \cdot s' + (a \bmod b) \cdot t'$, то тогда числа $s = t'$ и $t = s' - \lfloor a/b \rfloor \cdot t'$ удовлетворяют соотношению $d = a \cdot s + b \cdot t$.*

Доказательство. Заметим, что $a \bmod b = a - \lfloor a/b \rfloor \cdot b$, тогда

$$d = bs' + (a \bmod b)t = bs' + (a - \lfloor a/b \rfloor \cdot b)t' =$$

$$= at' + b(s' - \lfloor a/b \rfloor \cdot t') = a \cdot s + b \cdot t.$$

□

Здесь расширенный алгоритм Евклида использует те же самые шаги, что и простой алгоритм Евклида. Однако в каждом шаге мы выполняем три присваивания вместо одного. Алгоритм использует три набора переменных: r, s, t.

Алгоритм 4.2 Расширенный алгоритм Евклида

```
1:  procedure ExtendedEuclid(a, b)
2:      if b = 0 then
3:          return (a, 1, 0)
4:      else
5:          (d', s', t') ← ExtendedEuclid(b, a mod b)
6:          d ← d'
7:          s ← t'
8:          t ← s' − ⌊a/b⌋ · t'
9:          return (d, s, t)
10:     end if
11: end procedure
```

Рассмотрим выполнение алгоритма на примере заполнения таблицы 4.3. Пошаговое заполнение таблицы:

Таблица 4.3

Пример вычисления НОД(161, 28) по расширенному алгоритму Евклида

Номер строки	a	b	q	r	НОД	s	t
1	161	28	5	21	7	—1	6
2	28	21	1	7	7	1	—1
3	21	7	3	0	7	0	1
4	7	0	—	—	7	1	0

1. Заполняем исходные данные строка 1, столбцы a и b, т. е. $a = 161$, $b = 28$

2. Вычисляем в строке 1 столбцы q и r по формулам $q = \lfloor a/b \rfloor = 5$, $r = a \bmod b = 21$. Столбцы НОД, s, t оставляем пока пустыми.

3. Заполняем в строке 2 столбцы a и b. В столбец a записываем значение из столбца b предыдущей строки (т.е. значение 28), а в столбец b значение из предыдущей строки из столбца r равное 21.

4. Вычисляем в строке 2 столбцы q и r по формулам $q = \lfloor a/b \rfloor = 1$, $r = a \bmod b = 7$. Столбцы НОД, s, t оставляем пока пустыми.

5. Заполняем в строке 3 столбцы a и b. В столбец a записываем значение из столбца b предыдущей строки (т.е. значение 21), а в столбец b значение из предыдущей строки из столбца r, равное 7.

6. Вычисляем в строке 3 столбцы q и r по формулам $q = \lfloor a/b \rfloor = 3$, $r = a \bmod b = 0$. Столбцы НОД, s, t оставляем пока пустыми.

7. Заполняем в строке 4 столбцы a и b. В столбец a записываем значение из столбца b предыдущей строки (т.е. значение 7), а в столбец b значение из предыдущей строки из столбца r, равное 0.

8. Поскольку $b = 0$, то дальнейшие строчки не добавляем и переходим к заполнению столбцов НОД, s, t. Столбец НОД содержит одинаковое число — значение из столбца a последней строки в таблице, т.е. значение 7. Последняя строка всегда содержит в столбце s значение 1, а в столбце t значение 0.

9. Заполняем в строке 3 столбцы s, t. В столбец s всегда записываем значение из предыдущей строчки и столбца t, т.е. значение будет равно 0. В столбец t записываем значение, вычисленное по формуле $t = s' - q \cdot t'$, где s', t' — значения в предыдущей строки в соответствующих столбцах, т.е. $t = s' - q \cdot t' = 1 - 3 \cdot 0 = 1$.

10. Заполняем в строке 2 столбцы s, t аналогично предыдущей строке. В столбец s записываем значение из предыдущей строчки и столбца t, т.е. значение будет равно 1. В столбец t записываем значение $t = s' - q \cdot t' = 0 - 1 \cdot 1 = -1$.

11. Заполняем в строке 1 столбцы s, t аналогично предыдущей строке. В столбец s записываем значение из предыдущей строчки и столбца t, т.е. значение будет равно -1. В столбец t записываем значение $t = s' - q \cdot t' = 1 - 5 \cdot (-1) = 6$.

Упражнение 13. Используя расширенный алгоритм Евклида, найдите НОД, s, t для пар чисел (17, 0).

Упражнение 14. Используя расширенный алгоритм Евклида, найдите НОД, s, t для пар чисел (0, 45).

Упражнение 15. Используя расширенный алгоритм Евклида, найдите НОД, s, t для пар чисел (13, 1131).

Упражнение 16. Используя расширенный алгоритм Евклида, найдите НОД, s, t для пар чисел (18, 151).

4.3. Линейные диофантовы уравнения

Рассмотрим решение линейного диофантового уравнения

$$ax + by = c, \tag{4.1}$$

где $a, b, c \in \mathbb{Z}$ и $ab \neq 0$. Мы должны найти такие целые числа x и y, которые удовлетворяют уравнению (4.1). Пусть $d = \text{НОД}(a, b)$. Если $d \nmid c$, то левая часть делится на d, а правая не делится, что невозможно и,

следовательно, уравнение не имеет решения. Рассмотрим решение уравнения в случае, когда $d|c$. Поделим левую и правую часть уравнения (4.1) на d. Это возможно потому, что d делит a, b и c в соответствии с предположением. Получим эквивалентное уравнение

$$a_1 x + b_1 y = c_1,$$

где $a_1 = a/d$, $b_1 = b/d$, $c_1 = c/d$.

Найдем s и t в равенстве $a_1 s + b_1 t = 1$, используя расширенный алгоритм Евклида. Поскольку $d = \text{НОД}(a, b)$, то $\text{НОД}(a_1, b_1) = 1$. Тогда частное решение уравнения (4.1) можно записать как $x_0 = c_1 s$, $y_0 = c_1 t$. С помощью подстановки убеждаемся, что это действительно так. Общее решение может быть найдено по формулам: $x = x_0 + kb/d$, $y = y_0 + ka/d$, $k \in \mathbb{Z}$.

Пример 6. Найти частное и общее решения уравнения $21x + 14y = 35$.

Мы имеем $d = \text{НОД}(21, 14) = 7$. Т. к. $7|35$, следовательно, уравнение имеет бесконечное число решений. Мы можем разделить обе стороны уравнения на 7 и получим уравнение $3x + 2y = 5$. Используя расширенный алгоритм Евклида, мы находим s и t, такие, что $3s + 2t = 1$. Мы имеем $s = 1$ и $t = -1$, тогда $x_0 = 5 \cdot 1 = 5$ и $y_0 = 5 \cdot (-1) = -5$. Общее решение: $x = 5 + 2k$, $y = -5 - 3k$, где $k \in \mathbb{Z}$.

4.4. Обратные элементы

Когда мы работаем в модульной арифметике, нам часто нужно найти операцию, которая позволяет вычислить величину, обратную заданному числу.

4.4.1. Обратный элемент по сложению

В $\mathbb{Z}_n$ два числа a и a^{-1} противоположны (обратны по сложению), если

$$a + a^{-1} \equiv 0 \pmod{n}.$$

Распишем это сравнение через равенство

$$a + a^{-1} = 0 + q \cdot n,\ q \in \mathbb{Z}.$$

Выразим a^{-1}

$$a^{-1} = q \cdot n - a,\ q \in \mathbb{Z}.$$

Поскольку мы хотим, чтобы выполнялись неравенства $0 \leq a^{-1} \leq n - 1$, то нас устраивает единственный вариант $q = 1$ и

$$a^{-1} = n - a.$$

Например, обратный элемент по сложению для 4 в $\mathbb{Z}_{10}$ равен $10 - 4 = 6$.

Пример 7. Найдите все взаимно обратные пары по сложению в $\mathbb{Z}_{10}$.

Это следующие пары — (0, 0), (1, 9), (2, 8), (3, 7), (4, 6) и (5, 5). В этом списке 0 (также как и 5) — обратен самому себе.

4.4.2. Обратный элемент по умножению

В $\mathbb{Z}_n$ два числа a и a^{-1} обратны по умножению, если

$$a \cdot a^{-1} \equiv 1 \pmod{n}.$$

Например, если модуль равен 10, то для 3 обратный элемент по умножению будет 7. Другими словами, мы имеем $(3 \cdot 7) \bmod 10 = 1$. В модульной арифметике целое число может и не иметь обратного элемента по умножению. Найдем обратный элемент по умножению a^{-1}. По определению $a \cdot a^{-1} \equiv 1 \pmod{n}$. Распишем сравнение через равенство

$$a \cdot a^{-1} = 1 + q \cdot n,\ q \in \mathbb{Z}.$$

Перенесем все неизвестные влево

$$a \cdot a^{-1} - q \cdot n = 1. \tag{4.2}$$

Это линейное диофантово уравнение, которое имеет решение тогда и только тогда, когда НОД $(a,\ n) \mid 1$, т. е. НОД $(a,\ n) = 1$. Соответственно, обратный элемент по умножению существует, если НОД $(a,\ n) = 1$, и для его поиска можно применить расширенный алгоритм Евклида к линейному диофантовому уравнению (4.2). В этом уравнении a и n известны, а a^{-1}и q — неизвестны.

Пример 8. Найти обратный элемент 8 по умножению в $\mathbb{Z}_{10}$.

Обратный элемент по умножению не существует, потому что НОД $(8,\ 10) = 2 \nmid 1$. Другими словами, мы не можем найти число между 0 и 9, такое, что при умножении на 8 результат сравним с 1 по модулю 10. Более того, в $\mathbb{Z}_{10}$ есть только три пары, удовлетворяющие условиям существования обратного элемента по умножению: (1, 1), (3, 7) и (9, 9). Числа 0, 2, 4, 5, 6 и 8 не имеют обратных элементов по умножению.

Пример 9. Найти все мультипликативные обратные пары в $\mathbb{Z}_{11}$.

Мы имеем семь пар: (1, 1), (2, 6), (3, 4), (5, 9), (7, 8), (9, 9) и (10, 10). При переходе от $\mathbb{Z}_{10}$ к $\mathbb{Z}_{11}$ число пар увеличивается. Поскольку 11 взаимно просто со всеми числами от 1 до 10, то все числа от 1 до 10 имеют обратный элемент по умножению.

Пример 10. Найти обратный элемент 11 по умножению в $\mathbb{Z}_{26}$.

Мы используем таблицу 4.4, аналогичную одной из тех, которые мы уже применяли прежде при данных $n = 26$ и $a = 11$. Нас интересует

Таблица 4.4
Применение расширенного алгоритма Евклида для поиска обратного элемента по умножению

Номер строки	a	$b = n$	q	r	НОД	$s = a^{-1}$	$t = -q$
1	11	26	0	11	1	−7	3
2	26	11	2	4	1	3	−7
3	11	4	2	3	1	−1	3
4	4	3	1	1	1	1	−1
5	3	1	3	0	1	0	1
6	1	0	—	—	1	1	0

только значение a^{-1}. НОД $(26,\ 11) = 1$, что означает, что обратный элемент по умножению 11 существует. Расширенный алгоритм Евклида дает $a^{-1} = -7$. Обратный элемент равен $-7 \bmod 26 = 19$. Убедимся, что это действительно так $11 \cdot 19 \bmod 26 = 209 \bmod 26 = (26 \cdot 8 + 1) \bmod 26 = 1 \bmod 26$.

Упражнение 17. Найти обратный элемент 23 по умножению в $\mathbb{Z}_{100}$.

Упражнение 18. Найти обратный элемент 12 по умножению в $\mathbb{Z}_{26}$.

В дальнейшем мы будем использовать специальные обозначения для множеств. С помощью $\mathbb{Z}_n^*$ мы будем обозначать подмножество классов вычетов 1, ..., $n-1$, которые имеют обратные элементы по умножению по модулю n. Например, $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$, $\mathbb{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\}$, $\mathbb{Z}_6^* = \{1, 5\}$, $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$. Обратим внимание, что множества $\mathbb{Z}_7$ и $\mathbb{Z}_7^*$ различаются наличием только нулевого элемента.

4.5. Группы

Часто бывает удобно использовать алгебраические понятия в модульной арифметике. Напомним основные понятия.

Определение 10. Непустое множество G с заданной на нём бинарной операцией $* : G \times G \to G$ называется *группой* $\langle G, * \rangle$, если выполнены следующие аксиомы:

1. Ассоциативность $\forall a, b, c \in G : (a * b) * c = a * (b * c)$;
2. Существование нейтрального элемента $\exists \mathbf{1} \in G \quad \forall a \in G : \mathbf{1} * a = a * \mathbf{1} = a$;
3. Существование обратного элемента $\forall a \in G \quad \exists \left(a^{-1}\right) \in G : a * a^{-1} = a^{-1} * a = \mathbf{1}$.

В дальнейшем для обозначения групповой операции будем использовать знак «$*$», который в зависимости от группы может означать либо сложение, обозначаемое с помощью знака «+», либо умножение, обозначаемое

с помощью знака «·». В группах по сложению принято нейтральный элемент называть нулевым элементом или нулем ($\mathbf{0}$), а обратный элемент — противоположным ($-a$). В общем случае от группы не требуется выполнения свойства коммутативности ($\forall a, b \in G \quad a * b = b * a$), в группе $\langle \mathbb{Z}_n, + \rangle$, например, свойство коммутативности есть, такие группы называют *абелевыми*.

Определение 11. Непустое множество R с заданными на нём бинарными операциями $+ : R \times R \to R$ и $\cdot : R \times R \to R$ называется *кольцом* $\langle R, +, \cdot \rangle$, если $\langle R, + \rangle$ — абелева группа и выполнены следующие аксиомы:

1. Ассоциативность по умножению $\forall a, b, c \in R : (a \cdot b) \cdot c = a \cdot (b \cdot c)$;
2. Дистрибутивность умножения относительно сложения $\forall a, b, c \in R : a \cdot (b + c) = a \cdot b + a \cdot c$, $(b + c) \cdot a = b \cdot a + c \cdot a$.

Если $\forall a, b \in R \quad a \cdot b = b \cdot a$, то кольцо называют коммутативным кольцом. Если $\exists \mathbf{1} \in R$, $\mathbf{1} \neq \mathbf{0}$: $\forall a \in R \quad a \cdot \mathbf{1} = a$, то кольцо R называют кольцом с единицей.

Определение 12. Непустое множество F с заданными на нём бинарными операциями $+ : F \times F \to F$ и $\cdot : F \times F \to F$ называется *полем* $\langle F, +, \cdot \rangle$, если $\langle F, +, \cdot \rangle$ — коммутативное кольцо с единицей и существует обратный элемент $\forall a \in F$, $a \neq \mathbf{0}$ $\exists a^{-1} \in F$: $a \cdot a^{-1} = \mathbf{1}$.

Пример 11. Множество вычетов целых чисел с операцией сложения образуют группу $G = \langle \mathbb{Z}_n, + \rangle$. Проверим выполнение аксиом группы.

1. Замкнутость выполняется. Результат сложения двух вычетов в $\mathbb{Z}_n$ тоже вычет в $\mathbb{Z}_n$.
2. Ассоциативность выполняется. Результат $([a] + ([b] + [c])) \bmod n$ тот же самый, что в случае $(([a] + [b]) + [c]) \bmod n$.
3. Коммутативность выполняется. Мы имеем $([a] + [b]) \bmod n = ([b] + [a]) \bmod n$.
4. Существует нейтральный элемент. Мы имеем $([a] + [0]) \bmod n = ([0] + [a]) \bmod n = [a] \bmod n$.
5. Существует противоположный элемент — обратный элемент по сложению в $\mathbb{Z}_n$.

Пример 12. Множество ненулевых взаимно простых с n вычетов целых чисел с операцией умножения образуют группу $G = \langle \mathbb{Z}_n^*, \cdot \rangle$. Проверим выполнимость аксиом группы.

1. Пусть вычеты $[a], [b] \in \mathbb{Z}_n^*$, тогда НОД $(a, n) = 1$ и НОД $(b, n) = 1$. Следовательно, НОД $(ab, n) = 1$, т.е. $[ab] \in \mathbb{Z}_n^*$. Замкнутость выполняется.
2. Рассмотрим цепочку равенств с учетом ассоциативности операции умножения в целых числах $[a] \cdot ([b] \cdot [c]) = [a \cdot (b \cdot c) \bmod n] = [(a \cdot b) \cdot c \bmod n] = ([a] \cdot [b]) \cdot [c]$. Ассоциативность в G выполняется.

3. Аналогично выполняется коммутативность $[a]\cdot[b]=[a\cdot b \bmod n]=[b\cdot a \bmod n]=[b]\cdot[a]$

4. Для любого $[a]\in\mathbb{Z}_n^*$ существует нейтральный элемент $[1]\in\mathbb{Z}_n^*$. Мы имеем $[a]\cdot[1]=[a\cdot 1 \bmod n]=[1\cdot a \bmod n]=[1]\cdot[a]=[a]$.

5. Покажем, что для любого $[a]\in\mathbb{Z}_n^*$ существует обратный элемент. Пусть $\mathbb{Z}_n^*=\left\{[a_1],[a_2],...\left[a_{|\mathbb{Z}_n^*|}\right]\right\}$. Рассмотрим всевозможные произведения $[a]\cdot[a_i]$, $i=\overline{1,|\mathbb{Z}_n^*|}$. Убедимся, что все эти произведения попарно различны. Пусть это не так, тогда существуют $j,k\in\{1,2,...,|\mathbb{Z}_n^*|\}, j\neq k$ такие, что $[a]\cdot[a_j]=[a]\cdot[a_k]$, а следовательно, $aa_j\equiv aa_k \pmod n$, но поскольку $[a]\in\mathbb{Z}_n^*$ и НОД$(a,n)=1$, то $a_j\equiv a_k \pmod n$, что противоречит предположению $j\neq k$. Таким образом, все произведения $[a]\cdot[a_i]$, $i=\overline{1,|\mathbb{Z}_n^*|}$ различны, каждое из них принадлежит множеству $\mathbb{Z}_n^*$ в силу замкнутости операции, из существования нейтрального элемента в $\mathbb{Z}_n^*$ следует, что существует такое i, что $[a]\cdot[a_i]=[1]$. Итак, $[a_i]$ — обратный элемент для элемента $[a]$.

Определение 13. Подмножество $H\subseteq G$ называется *подгруппой* в группе G, если $\mathbf{1}\in H$; $h_1,h_2\in H\Rightarrow h_1h_2\in H$ и $h\in H\Rightarrow h^{-1}\in H$.

Определение 14. *Порядком группы* $\langle G,*\rangle$ называется мощность множества G (то есть число её элементов). Обозначается как $|G|$.

Если $|G|<\infty$, то группа называется конечной. Под возведением элемента a в степень m мы будем понимать m-кратное применение групповой операции к элементу, т.е.

$$a^m=e*\underbrace{a*a*...*a}_{m\text{ раз}}.$$

Определение 15. *Порядком элемента* g конечной группы $\langle G,*\rangle$ называется такое минимальное натуральное m, что $g^m=e$.

Справедлива теорема Лагранжа.

Теорема 4 (Лагранжа). *Если $\langle G,*\rangle$ — группа конечного порядка, то порядок любой её подгруппы H является делителем порядка группы.*

Следствие 1. *Порядок любого элемента делит порядок группы.*

Упражнение 19. Является ли группа $H=\langle\mathbb{Z}_{10},+\rangle$ подгруппой группы $G=\langle\mathbb{Z}_{12},+\rangle$?

Определение 16. Группа G называется *циклической группой* с образующим элементом $a\in G$, если любой элемент $g\in G$ может быть представлен как $g=a^n$, $n\in\mathbb{Z}$.

Определение 17. Подгруппа H группы G называется *циклической подгруппой* с образующим элементом $a\in H$, если любой элемент $h\in H$ может быть представлен как $h=a^n$, $n\in\mathbb{Z}$.

Пример 13. Группа $G = \langle \mathbb{Z}_6, + \rangle$ является циклической группой с образующим $a = 1$. В группе $G = \langle \mathbb{Z}_6, + \rangle$ существуют четыре циклических подгруппы (табл. 4.5). Это $H_1 = \langle \{0\}, + \rangle$, $H_2 = \langle \{0, 3\}, + \rangle$, $H_3 = \langle \{0, 2, 4\}, + \rangle$ и $H_4 = G$. Обратим внимание, что согласно теореме Лагранжа порядок подгрупп может быть только 1, 2, 3, 6. Для группы порядка, например, 17 подгруппы могут быть только порядка 1 и 17.

Упражнение 20. Является ли 5 образующим элементом в группе $G = \langle \mathbb{Z}_6, + \rangle$?

Упражнение 21. Покажите, что все вычеты $a^i \bmod n$, $i = \overline{1, order_n(a)}$ различны. Здесь через $order_n(a)$ обозначен порядок элемента a в группе $\langle \mathbb{Z}_n^*, \cdot \rangle$.

4.6. Линейное сравнение

В теории чисел и криптографии часто возникает задача отыскания решений сравнений первой степени вида

$$ax \equiv b \pmod{n}, \tag{4.3}$$

где $a, b \in \mathbb{Z}$, $n \nmid a$. Решение такого сравнения начинается с вычисления НОД$(a, n) = d$. При этом возможны два случая.

Таблица 4.5

Степени элементов в группе $\mathbb{Z}_6^+$

	a^0	a^1	a^2	a^3	a^4	a^5	a^6
$a = 0$	0	0	0	0	0	0	0
$a = 1$	0	1	2	3	4	5	0
$a = 2$	0	2	4	0	2	4	0
$a = 3$	0	3	0	3	0	3	0
$a = 4$	0	4	2	0	4	2	0
$a = 5$	0	5	4	3	2	1	0

1. Если b не кратно d, то у сравнения (4.3) нет решений.

2. Если b кратно d, то у сравнения d решений по модулю n. В этом случае в результате сокращения исходного сравнения на d получается сравнение

$$a_1 x \equiv b_1 \pmod{n_1},$$

где $a_1 = a/d$, $b_1 = b/d$, $n_1 = n/d$ являются целыми числами, причем a_1 и n_1 взаимно просты. Поэтому число a_1 можно обратить по модулю n_1, то есть найти такое число c, что $c \cdot a_1 \equiv 1 \pmod{n_1}$ (другими словами, $c \equiv a_1^{-1} \pmod{n_1}$). Теперь решение находится умножением полученного сравнения на c: $x \equiv ca_1 x \equiv cb_1 \equiv a_1^{-1} b_1 \pmod{n_1}$. Общее решение имеет вид $x = a_1^{-1} b_1 + kn_1$, $k = 0, 1, ...d - 1$.

Пример 14. Решить сравнение $10x \equiv 2 \pmod{15}$.

Найдем НОД (10, 15) = 5. Полученное число 5 не делится на 2, решение отсутствует.

Пример 15. Решить сравнение $14x \equiv 12 \pmod{18}$.

Заметим, что НОД $(14, 18) = 2$. Поскольку $2|12$, сравнение имеет решение. Сократим исходное сравнение на 2. Получим сравнение $7x \equiv 6 \pmod 9$. Найдем $7^{-1} \bmod 9 = 4$. Выпишем решение $x \equiv 6 \cdot 4 \equiv 24 \equiv 6 \pmod 9$. Получим два решения: $x_1 = 6 + 0 \cdot 9 + 18k = 6 + 18k$, $k \in \mathbb{Z}$ и $x_2 = 6 + 1 \cdot 9 + 18k = 15 + 18k$, $k \in \mathbb{Z}$.

Пример 16. Решить сравнение $3x + 4 \equiv 6 \pmod{13}$.

Сначала приведем сравнение к форме $a \cdot x \equiv b \pmod n$. Мы прибавляем -4 к обеим сторонам сравнения. Получим $3x \equiv 2 \pmod{13}$. Поскольку НОД $(3, 13) = 1$, сравнение имеет только одно решение, $x = 2 \cdot 3^{-1} \bmod 13 + 13k = 2 \cdot 9 \bmod 13 + 13k = 5 + 13k$, $k \in \mathbb{Z}$.

4.7. Системы линейных сравнений

Для решения системы линейных сравнений используется китайская теорема об остатках.

Теорема 5 (Китайская теорема об остатках)**.** *Если натуральные числа $a_1, a_2, \ldots, a_n$ попарно взаимно просты, то для любых целых $r_1, r_2, \ldots, r_n$ таких, что $0 \leqslant r_i < a_i$ при всех $i \in \{1, 2, \ldots, n\}$, найдётся число N, которое при делении на a_i даёт остаток r_i при всех $i \in \{1, 2, \ldots, n\}$. Более того, если найдутся два таких числа N_1 и N_2, то $N_1 \equiv N_2 \pmod{a_1 \cdot a_2 \cdot \ldots \cdot a_n}$.*

Доказательство. Рассмотрим систему уравнений:

$$\begin{cases} x & \equiv r_1 \pmod{a_1}; \\ x & \equiv r_2 \pmod{a_2}; \\ & \vdots \\ x & \equiv r_n \pmod{a_n}. \end{cases} \tag{4.4}$$

Если наборы $(r_1, r_2, \ldots, r_n)$ и $(a_1, a_2, \ldots, a_n)$ удовлетворяют условию теоремы, то решение системы (4.4) существует и единственно с точностью до операции взятия по модулю M, где $M = \prod_{i=1}^{n} a_i$, причем справедлива формула

$$x = \sum_{i=1}^{n} r_i M_i M_i^{-1}, \tag{4.5}$$

где $M_i = \frac{M}{a_i}$, а M_i^{-1} — обратный элемент по умножению по модулю a_i.

Покажем, что определенный таким образом x является решением. Проверим, что для него выполняется i-е сравнение в системе:

$$x = \sum_{j=1}^{n} r_j M_j M_j^{-1} \equiv r_i M_i M_i^{-1} \equiv r_i \pmod{a_i}.$$

Второе равенство справедливо, т. к. $M_j \equiv \prod_{k\neq j}^{n} a_k \equiv 0 \pmod{a_i}$ при всех $i \neq j$. Третье сравнение также выполняется, поскольку M_i^{-1} является обратным для M_i по модулю a_i. Повторяя рассуждения для всех i убедимся, что x, определенный формулой (4.5), является решением для (4.4). В силу выбранного числа M все числа $x' \equiv x \pmod{M}$ будут удовлетворять системе.

Пусть $A = \{0, 1, \ldots, M-1\}$ и $B = \left\{(r_1, r_2, \ldots, r_n) | r_i \in \mathbb{Z}_{a_i},\ i = \overline{1,n}\right\}$, тогда покажем, что среди элементов множества A не найдется другого решения, кроме найденного нами ранее. Предположим, что получилось найти хотя бы два решения $x_1, x_2 \in A$ для некоторого набора остатков $r = (r_1, r_2, \ldots, r_n)$. Так как множество B является равномощным множеству A, то для $\overline{A}_x := A \setminus \{x_1, x_2\}$и $\overline{B}_r := B \setminus \{r\}$ выполнено $|\overline{A}_x| < |\overline{B}_r|$. Однако по доказанному ранее для любого набора из $\overline{B}_r$ существует решение из $\overline{A}_x$, следовательно, по принципу Дирихле, найдутся как минимум два набора остатков, которым соответствует одно и то же $x \in A$. Для такого x найдется a_i такое, что $x \equiv r_1 \pmod{a_i}$ и $x \equiv r_2 \pmod{a_i}$, где $r_1 \neq r_2$. Получили противоречие. □

Пример 17. Найдите решение системы уравнений

$$\begin{cases} x \equiv 2 \pmod 3; \\ x \equiv 3 \pmod 5; \\ x \equiv 2 \pmod 7. \end{cases}$$

Найдем $M = \prod_{i=1}^{n} a_i = 3 \cdot 5 \cdot 7 = 105$. Следовательно, $M_1 = \frac{M}{a_1} = \frac{105}{3} = 35$. Аналогично, $M_2 = 21$, $M_3 = 15$. Найдем обратные элементы: $M_1^{-1} = 35^{-1} \pmod 3 = 2^{-1} \pmod 3 = 2$, $M_2^{-1} = 21^{-1} \pmod 5 = 1^{-1} \pmod 3 = 1$, $M_3^{-1} = 15^{-1} \pmod 7 = 1^{-1} \pmod 7 = 1$. Можем вычислить значение x по формуле (4.5), т.е. $x = (2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1) \bmod 105 = 23 \bmod 105$. Ответ: $x = 23 + 105k$, $k \in \mathbb{Z}$.

Упражнение 22. Найти целое, которое дает в остатке 3, если его разделить на 7 и 13, но без остатка делится на 12.

4.8. $\varphi-$функция Эйлера

Определение 18. Для любого $n \in \mathbb{N}$ *функция Эйлера* $\varphi(n)$ определяется как число натуральных чисел не больших n и взаимно простых с n.

Функция Эйлера обладает следующими свойствами:

1. $\varphi(1) = 1$;
2. $\varphi(p) = p - 1$, если p — простое число;
3. $\varphi(mn) = \varphi(m)\varphi(n)$, если m, n — взаимно простые числа;

4. $\varphi(p^t) = p^t - p^{t-1}$, если p — простое число;

Обобщая предыдущие свойства можно записать

$$\varphi(n) = \left(p_1^{t_1} - p_1^{t_1-1}\right)\left(p_2^{t_2} - p_2^{t_2-1}\right)\ldots\left(p_k^{t_k} - p_k^{t_k-1}\right),$$

где $n = p_1^{t_1}p_2^{t_2}\ldots p_k^{t_k}$ и p_i, $i = \overline{1,k}$ — различные простые числа.

Свойства 1 и 2 проверяются непосредственно.

Теорема 6. *$\varphi(mn) = \varphi(m)\varphi(n)$, если m, n — взаимно простые числа.*

Доказательство. Рассмотрим числа

$$V(x,\, y) = mx + ny, \tag{4.6}$$

при $x = \overline{1,\, n}$, $y = \overline{1,m}$.

Покажем, что все числа $V(x,y)$ дают разные остатки по модулю mn. Предположим, что это не так, т.е. существуют такие различные пары (x_1, y_1) и (x_2, y_2), для которых $mx_1 + ny_1 \equiv mx_2 + ny_2 \pmod{mn}$, тогда $m(x_1 - x_2) + n(y_1 - y_2) \equiv 0 \pmod{mn}$. Первое слагаемое делится на m, значит, и второе слагаемое делится на m. В силу взаимной простоты m и n получаем, что $y_1 - y_2$ делится на m, что возможно только при $y_1 = y_2$. Значит, $m(x_1 - x_2) \equiv 0 \pmod{ab}$, откуда следует, что $x_1 - x_2$ делится на n, что бывает только при $x_1 = x_2$. Таким образом, наше предположение оказалось неверным, и это означает, что числа $V(x,\, y) = mx + ny$ при $x = \overline{1,\, n}$, $y = \overline{1,m}$ дают различные остатки по модулю mn.

Осталось заметить, что $V(x,\, y)$ взаимно просто с mn тогда и только тогда, когда x взаимно прост с n, а y взаимно прост с m. Действительно, пусть $V(x,\, y)$ взаимно просто с mn, но (для определённости) x имеет общий с n делитель $d > 1$. Но тогда d делит и первое и второе слагаемое в 4.6, значит $d \mid V(x,\, y)$, что противоречит предположению.

Обратно, пусть НОД $(x,\, n) = 1$ и НОД $(y,\, m) = 1$, но mn имеет общий с $V(x,\, y)$ делитель $d > 1$. Пусть p — произвольный простой делитель числа d. Тогда $p \mid nm$. Пусть (для определённости) $p \mid m$. Тогда p не делит y (в силу взаимной простоты m и y). Но $p \mid ny$ (так как $p \mid (mx + ny)$). Значит, $p \mid n$. Но тогда p — общий делитель m и n, что противоречит взаимной простоте m и n.

Остатков по модулю mn, взаимно простых с mn, ровно $\varphi(mn)$ штук; чисел x, взаимно простых с n, ровно $\varphi(n)$; чисел y, взаимно простых с m, ровно $\varphi(m)$. Значит, $\varphi(mn) = \varphi(m)\varphi(n)$. □

Предложение 8. *$\varphi(p^t) = p^t - p^{t-1}$, если p — простое число.*

Доказательство. Рассмотрим числа от 1 до $p^t - 1$. Среди этих чисел есть числа, которые имеют общие делители с p^t. Эти числа записываются, как $1p$, $2p$, $3p$, $\ldots$, $p^t - p$ и все они кратны p. Всего таких чисел $p^{t-1} - 1$. Все

остальные числа взаимно просты с p^t, и их количество выражается как $(p^t - 1) - \left(p^{t-1} - 1\right) = p^t - p^{t-1}$. □

Необходимо отметить, что значение $\varphi(n)$ для больших чисел может быть найдено, если известно разложение n на простые множители. Это означает, что сложность нахождения $\varphi(n)$ зависит от сложность нахождения разложения n.

Пример 18. Вычислить $\varphi(13)$.

Поскольку 13 — простое число, $\varphi(13) = 13 - 1 = 12$.

Пример 19. Вычислить $\varphi(10)$.

Мы можем использовать третье правило: $\varphi(10) = \varphi(2) \cdot \varphi(5) = (2 - 1) \cdot (5 - 1) = 1 \cdot 4 = 4$, поскольку 2 и 5 взаимно простые числа.

Пример 20. Вычислить $\varphi(240)$.

Мы можем записать $240 = 10 \cdot 3 \cdot 8 = 5 \cdot 3 \cdot 16 = 2^4 \cdot 3^1 \cdot 5^1$. Тогда $\varphi(240) = (2^4 - 2^3) \cdot (3^1 - 3^0) \cdot (5^1 - 5^0) = 8 \cdot 2 \cdot 4 = 64$

Пример 21. Вычислить $\varphi(49)$.

Поскольку $49 = 7^2$ мы не можем использовать свойство 2, т.к. там требуется, чтобы множители были взаимно простыми, но можно применить свойство 3. Итак, $\varphi(49) = \varphi(7^2) = 7^2 - 7^1 = 49 - 7 = 42$.

Пример 22. Сколько элементов в группе $\langle \mathbb{Z}_{14}^*, \cdot \rangle$?

В группу $\langle \mathbb{Z}_{14}^*, \cdot \rangle$ входят все натуральные числа меньшие 14 и взаимно простые с 14. Т.е. количество элементов в группе $\varphi(14) = \varphi(2) \cdot \varphi(7) = 1 \cdot 6 = 6$. Это элементы 1, 3, 5, 9, 11, 13.

Упражнение 23. Доказать, что если $n > 2$, то значение $\varphi(n)$ — четное.

4.9. Алгоритм быстрого возведения в степень по модулю

Большинство операций в кольце вычетов $\mathbb{Z}_n$ можно выполнять, выполнив сначала действия с числами, а затем находя остаток от деления результата на модуль n. Однако с операцией возведения в степень такой порядок является очень неэффективным. Например, если мы захотим вычислить $2^{199} \bmod 1003$ с помощью обычного калькулятора, то результат окажется неверным, т. к. при вычислении 2^{199} разрядная сетка будет переполнена. В то же время эту же операции несложно выполнить с помощью алгоритма быстрого возведения в степень по модулю заданного натурального числа.

Предположим, что требуется вычислить $z = a^b \bmod n$. Рассмотрим следующий способ. Представим b в двоичный системе исчисления: $b =$

$(b_k b_{k-1} ... b_1 b_0)_2$, $b_i \in \{0, 1\}$. В нашем примере $199 = 11000111_2$. Заметим, что $a^b \bmod n$ можно представить как

$$a^b \bmod n = a^{b_k \cdot 2^k + b_{k-1} \cdot 2^{k-1} + .. + b_0 \cdot 2^0} \bmod n =$$

$$= a^{b_k \cdot 2^k} \cdot a^{b_{k-1} \cdot 2^{k-1}} \cdot ... \cdot a^{b_0 \cdot 2^0} \bmod n = a^{b_k \cdot 2^k} \cdot a^{b_{k-1} \cdot 2^{k-1}} \cdot ... \cdot a^{b_0 \cdot 2^0} \bmod n =$$

$$= \left(a^{2^k}\right)^{b_k} \cdot \left(a^{2^{k-1}}\right)^{b_{k-1}} \cdot ... \cdot \left(a^{2^0}\right)^{b_0} \bmod n = \prod_{i=0}^{k} \left(a^{2^i}\right)^{b_i} \bmod n,$$

где $a^{2^{i+1}} = \left(a^{2^i}\right)^2$. Это позволяет воспользоваться таблицей 4.6. Заполним её. Сначала заполним первую строку — это двоичное представление числа b. Затем заполним вторую строку, начнем с самой правой колонки, туда запишем a, в следующую колонку запишем значение из правой колонки, возведенное в квадрат по модулю n. Заполним третью строку, путем возведения значения из второй строки в степень b_i. Для получения окончательного результата требуется перемножить по модулю n все значения из третьей строки. В нашем случае $2^{199} \bmod 1003 = 851 \cdot 477 \cdot 1 \cdot 1 \cdot 1 \cdot 16 \cdot 4 \cdot 2 \bmod 1003 = 247$.

Таблица 4.6

Таблица для быстрого возведения в степень

b_i	b_k	...	b_1	b_0
a^{2^i}	$a^{2^k} \bmod n$	...	$a^{2^1} \bmod n$	$a^{2^0} \bmod n$
$\left(a^{2^i}\right)^{b_i}$	$\left(a^{2^k}\right)^{b_k} \bmod n$		$\left(a^{2^1}\right)^{b_1} \bmod n$	$\left(a^{2^0}\right)^{b_0} \bmod n$

Таблица 4.7

Таблица для быстрого возведения в степень для вычисления $2^{199} \bmod 1003$

b_i	1	1	0	0	0	1	1	1
a^{2^i}	851	477	936	341	256	16	4	2
$\left(a^{2^i}\right)^{b_i}$	851	477	1	1	1	16	4	2

Вариант этого алгоритма без использования таблицы представлен алгоритме 4.3. Здесь таблица строится по столбцам и в каждый момент времени хранится только текущий столбец. Время работы алгоритма 4.3 прямо пропорционально количеству бит в двоичной записи числа b.

Упражнение 24. Продемонстрируйте работу алгоритма 4.3 при вычислении $7^{53} \bmod 29$.

Алгоритм 4.3 Алгоритм быстрого возведения в степень

```
1:  procedure FastPow(a, b, n)        ▷ Алгоритм вычисляет a^b mod n
2:      c ← 1
3:      while b ≠ 0 do
4:          if b mod 2 = 0 then
5:              b ← b/2
6:              a ← (a · a) mod n
7:          else
8:              b ← b − 1
9:              c ← (c · a) mod n
10:         end if
11:     end while
12:     return c
13: end procedure
```

4.10. Малая теорема Ферма

Малая теорема Ферма играет очень важную роль в теории чисел и криптографии.

Теорема 7 (Малая теорема Ферма)**.** *Если p — простое число и a — целое число, такое, что p не является делителем a, то $a^{p-1} \equiv 1 \pmod{p}$.*

Также иногда полезна бывает альтернативная формулировка.

Теорема 8. *Если p — простое число и a — целое число, то $a^p \equiv a \pmod{p}$.*

Доказательство. Докажем, что для любого простого p и целого неотрицательного a, $a^p - a$ делится на p. Доказываем индукцией по a.

База. Для $a = 0$, $a^p - a = 0$ и делится на p.

Индукционный шаг. Пусть утверждение верно для $a = k$. Докажем его для $a = k + 1$. Выпишем $a^p - a$

$$a^p - a = (k+1)^p - (k+1) =$$

$$= k^p + 1 + \sum_{l=1}^{p-1} C_p^l k^l - k - 1 = k^p - k + \sum_{l=1}^{p-1} k^l C_p^l,$$

здесь $k^p - k$ делится на p по предположению индукции. Что же касается остальных слагаемых, то

$$C_p^l = \frac{p!}{l!(p-l)!}.$$

Для $1 \leq l \leq p-1$ числитель этой дроби делится на p, а знаменатель — взаимно прост с p (p — простое), следовательно, C_p^l делится на p. Таким образом, вся сумма

$$k^p - k + \sum_{l=1}^{p-1} k^l C_p^l$$

делится на p. Случай, когда a отрицательное, рассматривается в упражнении 25. □

Эту теорему можно рассматривать, как частный случай теоремы Лагранжа. Действительно, при простом p множество ненулевых элементов кольца $\mathbb{Z}_p^*$ образует группу по умножению, имеющую $p-1$ элемент. По теореме Лагранжа порядок любого элемента $a \in \mathbb{Z}_p^*$ является делителем порядка $p-1$, откуда $a^{p-1} \equiv 1 \pmod{p}$.

Малая теорема Ферма позволяет сократить вычисления при возведении чисел по модулю p в степень больше, чем $p-2$. Следующие примеры показывают это.

Пример 23. Найдите результат $6^{10} \bmod 11$.

Мы имеем $6^{10} \bmod 11 = 1$. Это первая версия малой теоремы Ферма, где $p = 11$.

Пример 24. Найдите результат $3^{12} \bmod 11$.

Здесь степень 12 и модуль 11 не соответствуют условиям теоремы Ферма. Но, применяя преобразования, мы можем привести решение к использованию малой теоремы Ферма.

$$3^{12} \bmod 11 = \left(3^{10} \cdot 3^2\right) \bmod 11 =$$
$$= \left(\left(3^{10} \bmod 11\right) \cdot \left(3^2 \bmod 11\right)\right) \bmod 11 = 9 \bmod 11 = 9.$$

Если p — простое число и a — целое число, такое, что p не является его делителем, тогда $a^{-1} \bmod p = a^{p-2} \bmod p$. Это может быть доказано, если мы умножим обе стороны равенства на a и используем первую версию малой теоремы Ферма. Это позволяет не применять расширенный алгоритм Евклида для нахождения обратных элементов по умножению.

Пример 25. Найти обратные элементы по умножению $8^{-1} \bmod 17$, $5^{-1} \bmod 23$, $6^{-1} \bmod 101$, $22^{-1} \bmod 211$.

Применяем малую теорему Ферма и получаем:
1. $8^{-1} \bmod 17 = 8^{17-2} \bmod 17 = 8^{15} \bmod 17 = 15$.
2. $5^{-1} \bmod 23 = 5^{23-2} \bmod 23 = 5^{21} \bmod 23 = 14$.
3. $6^{-1} \bmod 101 = 6^{101-2} \bmod 101 = 6^{99} \bmod 101 = 32$.
4. $22^{-1} \bmod 211 = 22^{211-2} \bmod 211 = 22^{209} \bmod 211 = 48$.

Для возведения в большую степень используем алгоритм быстрого возведения в степень (алг. 4.3).

Из теоремы Ферма сразу следует, что если для некоторого $a < p$ выполнено условие $a^{p-1} \not\equiv 1 \pmod{p}$, тогда число p является составным. Однако обращение малой теоремы Ферма не верно – существуют составные числа p, для которых выполняется условие Ферма для каждого a, не сравнимого с p. Такие числа называются числами Кармайкла.

Упражнение 25. Докажите малую теорему Ферма для отрицательных значений a.

Упражнение 26. Докажите, что формулировки малой теорема Ферма эквивалентны.

4.11. Теорема Эйлера

Малая теорема Ферма является частным случаем теоремы Эйлера. Если модуль в малой теореме Ферма — простое число, то в теореме Эйлера модуль может быть и составным числом.

Теорема 9 (Эйлера). *Если a и n — взаимно простые, то*

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

Доказательство. Пусть $x_1, \ldots, x_{\varphi(n)}$ — все различные натуральные числа, меньшие n и взаимно простые с ним. Рассмотрим всевозможные произведения $x_i a$ для всех i от 1 до $\varphi(n)$. Поскольку a взаимно просто с n и x_i взаимно просто с n, то и $x_i a$ также взаимно просто с n, то есть $x_i a \equiv x_j \pmod{n}$ для некоторого j.

Заметим, что все остатки $x_i a$ при делении на n различны. Действительно, пусть это не так, то существуют такие $i \neq j$, что $x_i a \equiv x_j a \pmod{n}$ или $(x_i - x_j)a \equiv 0 \pmod{n}$. Так как a взаимно просто с n, то последнее равенство равносильно тому, что $x_i - x_j \equiv 0 \pmod{n}$ или $x_i \equiv x_j \pmod{n}$. Это противоречит тому, что числа $x_1, \ldots, x_{\varphi(n)}$ попарно различны по модулю n.

Перемножим все сравнения вида $x_i a \equiv x_j \pmod{n}$. Получим $x_1 \cdot \ldots \cdot x_{\varphi(n)} a^{\varphi(n)} \equiv x_1 \cdot \ldots \cdot x_{\varphi(n)} \pmod{n}$ или $x_1 \cdot \ldots \cdot x_{\varphi(n)}(a^{\varphi(n)} - 1) \equiv 0 \pmod{n}$. Так как число $x_1 \cdot \ldots \cdot x_{\varphi(n)}$ взаимно просто с n, то последнее сравнение равносильно тому, что $a^{\varphi(n)} - 1 \equiv 0 \pmod{n}$ или $a^{\varphi(n)} \equiv 1 \pmod{n}$. □

Аналогично малой теореме Ферма теорему Эйлера можно доказать через теорему Лагранжа. Рассмотрим мультипликативную группу $\mathbb{Z}_n^*$ обратимых элементов кольца вычетов $\mathbb{Z}_n$. Её порядок равен $\varphi(n)$ согласно определению функции Эйлера. Поскольку число $a \in \mathbb{Z}$ взаимно просто с

n, соответствующий ему элемент $[a]$ в $\mathbb{Z}_n$ является обратимым и принадлежит $\mathbb{Z}_n^*$. Элемент $[a] \in \mathbb{Z}_n^*$ порождает циклическую подгруппу, порядок которой, согласно теореме Лагранжа, делит $\varphi(n)$, отсюда $[a]^{\varphi(n)} = [1]$.

Теорема Эйлера может применяться, чтобы найти обратные элементы по умножению по составному модулю. Если n и a — взаимно простые, то $a^{-1} \bmod n = a^{\varphi(n)-1} \bmod n$. Это может быть легко доказано умножением обеих сторон равенства на a.

Пример 26. Обратные элементы по умножению по составному модулю могут быть найден без использования расширенного алгоритма Евклида, если мы знаем разложение на множители составного числа:

$8^{-1} \bmod 77 = 8^{\varphi(77)-1} \bmod 77 = 8^{59} \bmod 77 = 29$.

$7^{-1} \bmod 15 = 7^{\varphi(15)-1} \bmod 15 = 7^7 \bmod 15 = 13$.

$6^{-1} \bmod 187 = 6^{\varphi(187)-1} \bmod 187 = 6^{159} \bmod 187 = 53$.

$71^{-1} \bmod 100 = 7^{\varphi(100)-1} \bmod 100 = 7^{39} \bmod 100 = 31$.

4.12. Квадратичные сравнения

Общий вид сравнения второй степени по простому модулю p имеет вид

$$c_0 x^2 + c_1 x + c_2 \equiv 0 \pmod{p}, \tag{4.7}$$

где c_0, c_1, $c_2 \in \mathbb{Z}$, $n \nmid c_0$. Поиск решения сравнения (4.7) при $p = 2$ не представляет труда, так как в этом случае оно принадлежит классу вычетов $[0]$ или $[1]$. Изучать (4.7) будем в предположениях, что модуль $p > 2$ — простое нечетное число, которое не делит коэффициент c_0. Если $p|c_0$, то сравнение второй степени эквивалентно сравнению первой степени $c_1 x + c_2 \equiv 0 \pmod{p}$.

Теорема 10. *Если $p > 2$ и p не делит c_0, то сравнение (4.7) эквивалентно сравнению вида*

$$(x + c)^2 \equiv a \pmod{p}.$$

Доказательство. Сравнение (4.7) (если p не делит c_0) можно сначала привести к виду $x^2 + b_1 x + b_2 \equiv 0 \pmod{p}$, умножая на левую и правую часть на $c_0^{-1} \pmod{p}$. Рассмотрим два возможных случая. Если $2|b_1$, то сравнение (4.7) можно записать в виде

$$\left(x + \frac{b_1}{2}\right)^2 \equiv \left(\frac{b_1}{2}\right)^2 - b_2 \pmod{p}.$$

Если $2 \nmid b_1$, то сравнение (4.7) заменяем эквивалентным сравнением

$$x^2 + (b_1 + p)x + b_2 \equiv 0 \pmod{p},$$

которое, поскольку p нечетно и, следовательно, $2|(b+p)$, можно записать в виде

$$\left(x+\frac{b_1+p}{2}\right)^2 \equiv \left(\frac{b_1+p}{2}\right)^2 - b_2 \pmod{p}.$$

□

Если положить $x+c=z$, то исследование сравнения (4.7) сводится к исследованию сравнения $x^2 \equiv a \pmod{p}$. Итак, рассмотрим сравнение

$$x^2 \equiv a \pmod{p}. \tag{4.8}$$

Будем считать, что модуль p не делит a, так как в этом случае сравнение (4.8) имеет только одно решение $x \equiv 0 \pmod{p}$. Если p не делит a, то сравнение (4.8) может не иметь решений. Например, легко проверить, что сравнению $x^2 \equiv 2 \pmod{5}$ не удовлетворяет ни один из классов по модулю 5.

Теорема 11. *Сравнение (4.8) по простому модулю может иметь не более двух решений.*

Таким образом, если исключить из рассмотрения значения a, кратные p, то сравнение (4.8) либо имеет два решения, либо не имеет ни одного решения. Если при некотором a сравнение (4.8) имеет два решения, то и для любого $b \equiv a \pmod{p}$ это сравнение имеет два решения, поэтому естественно рассматривать сравнение (4.8) не для отдельных чисел, а для соответствующих классов.

Определение 19. Класс чисел по модулю p называется *классом квадратичных вычетов* по этому модулю, если для чисел a, принадлежащих этому классу, сравнение $x^2 \equiv a \pmod{p}$ имеет два решения.

Определение 20. Класс чисел по модулю p называется *классом квадратичных невычетов*, если для чисел a, принадлежащих этому классу, сравнение $x^2 = a \pmod{p}$ не имеет решений.

Все числа, принадлежащие классам квадратичных вычетов, т. е. все числа a, для которых сравнение $x^2 \equiv a \pmod{p}$ имеет два решения, будем называть квадратичными вычетами по модулю p и аналогично все числа a, для которых это сравнение не имеет решений, — квадратичными невычетами по этому модулю. Числа a, принадлежащие классу $[0]$, для которых сравнение (4.8) имеет одно решение, не причисляются ни к квадратичным вычетам, ни к квадратичным невычетам.

Теорема 12. *По любому простому модулю $p > 2$ число классов квадратичных вычетов равно числу классов квадратичных невычетов.*

Теорема 13 (Критерий Эйлера). *Число a, не делящееся на простое число p, $(p > 2)$, является квадратичным вычетом тогда и только тогда, когда*

$$a^{(p-1)/2} \equiv 1 \pmod{p} \tag{4.9}$$

и является квадратичным невычетом тогда и только тогда, когда

$$a^{(p-1)/2} \equiv -1 \pmod{p}. \tag{4.10}$$

Доказательство. Если a — квадратичный вычет по модулю p, то существует такое число c, что $a \equiv c^2 \pmod{p}$. Возводя обе части последнего сравнения в степень $(p-1)/2$, получаем (с учетом малой теоремы Ферма) $a^{(p-1)/2} \equiv c^{p-1} \equiv 1 \pmod{p}$.

Если a — квадратичный невычет по модулю p, тогда существует такое число j, что $a = g^{2j+1}$, где g — образующий элемент группы $\mathbb{Z}_p^*$. Вычислим чему равно

$$a^{\frac{p-1}{2}} \equiv g^{\frac{(2j+1)(p-1)}{2}} \equiv g^{\frac{2jp-2j+p-1}{2}} \equiv g^{j(p-1)+\frac{p-1}{2}} \equiv g^{\frac{p-1}{2}} \pmod{p}. \tag{4.11}$$

Обозначим правую часть как $g^{\frac{p-1}{2}} \equiv x \pmod{p}$, и возведем обе части сравнения в квадрат $x^2 \equiv g^{p-1} \equiv 1 \pmod{p}$. Последнее сравнение может иметь два решения $x \equiv \pm 1 \pmod{p}$, но поскольку g — образующий элемент группы, то только $p-1$ степень элемента g равна 1. Следовательно, $g^{\frac{p-1}{2}} \equiv -1 \pmod{p}$ и, с учётом (4.11), $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.

Согласно малой теореме Ферма имеем $a^{p-1} \equiv 1 \pmod{p}$, тогда

$$a^{p-1} - 1 \equiv (a^{(p-1)/2} - 1)(a^{(p-1)/2} + 1) \equiv 0 \pmod{p}. \tag{4.12}$$

Оба сомножителя в последнем сравнении на модуль p делиться не могут, так как их разность, которая равна 2, тоже бы делилась на p. Таким образом, для каждого целого a $(1 < a < p)$ обязательно выполняется только одно из сравнений

$$a^{(p-1)/2} \equiv 1 \pmod{p} \text{ или } a^{(p-1)/2} \equiv -1 \pmod{p}.$$

По условиям теоремы $p \nmid a$, и значит a является либо квадратичным вычетом, либо квадратичным невычетом, следовательно, если $a^{(p-1)/2} \equiv 1 \pmod{p}$, то a является квадратичным вычетом, если $a^{(p-1)/2} \equiv -1 \pmod{p}$, то a является квадратичным невычетом. □

Пример 27. Число 2 — квадратичный невычет по простому модулю 11, так как $2^5 \equiv 32 \equiv -1 \pmod{11}$. Число 3 — квадратичный вычет по модулю 11, так как $3^5 \equiv 243 \equiv 1 \pmod{11}$.

Теорема 14. *Числа 1^2, 2^2, ..., $\left(\frac{p-1}{2}\right)^2$ образуют систему представителей всех классов квадратичных вычетов по простому модулю $p > 2$.*

Пример 28. Рассмотрим число $p = 7$. Тогда

$$1^2 = 1 \equiv 1 \bmod 7, \quad 2^2 = 4 \equiv 4 \bmod 7, \quad 3^2 = 9 \equiv 2 \bmod 7,$$
$$4^2 = 16 \equiv 2 \bmod 7 \quad 5^2 = 25 \equiv 4 \bmod 7, \quad 6^2 = 36 \equiv 1 \bmod 7.$$

Из приведенных соотношений видно, что для $p = 7$ только числа 1, 2, 4 выступают представителями класса квадратичных вычетов. Эти вычеты определяются уже для чисел $1^2, 2^2, 3^2$. Заметим, что числа 3, 5, 6 являются квадратичными невычетами, так что сравнения

$$x^2 \equiv 3 \pmod 7, \quad x^2 \equiv 5 \pmod 7, \quad x^2 \equiv 6 \pmod 7.$$

не имеют решения.

Чтобы найти решение квадратичного сравнения $x^2 \equiv a \pmod p$, заметим, что простое число ($p \neq 2$) может быть представлено либо как $p = 4k + 1$, либо как $p = 4k + 3$, где $k \in \mathbb{Z}_+$. Если $p \equiv 3 \pmod 4$, то известна следующая теорема.

Теорема 15. *Если $p = 4k + 3$, $k \in \mathbb{N} \cup \{0\}$ и a — квадратичный вычет в $\mathbb{Z}_p^*$, то $x \equiv \pm a^{\frac{p+1}{4}} \pmod p$ являются решениями сравнения (4.8).*

Доказательство. Проверим выполнение сравнения (4.8) при заданном x. Применим критерий Эйлера:

$$x^2 \equiv \left(a^{\frac{p+1}{4}}\right)^2 \equiv a^{\frac{p+1}{2}} \equiv a^{\frac{p-1+2}{2}} \equiv a^{\frac{p-1}{2}+1} \equiv a^{\frac{p-1}{2}} a \equiv 1a \equiv a \pmod p,$$

$$x^2 \equiv \left(-a^{\frac{p+1}{4}}\right)^2 \equiv a^{\frac{p+1}{2}} \equiv a^{\frac{p-1+2}{2}} \equiv a^{\frac{p-1}{2}+1} \equiv a^{\frac{p-1}{2}} a \equiv 1a \equiv a \pmod p.$$

□

Если n составное число, то для решения сравнения $x^2 \equiv a \pmod n$ необходимо разложить n на простые множители и применить китайскую теорему об остатках. Например, для решения сравнения

$$x^2 \equiv a \pmod n, \; n = p \cdot q, \tag{4.13}$$

где p и q простые, необходимо найти p и q, составить два квадратичных сравнения $x^2 \equiv a \pmod p$ и $x^2 \equiv a \pmod q$, найти их решения. Обозначим найденные решения $\pm x_p$, $\pm x_q$. А дальше решить четыре системы сравнений с помощью китайской теоремы об остатках:

$$\begin{cases} x_1 \equiv +x_p \pmod p, \\ x_1 \equiv +x_q \pmod q, \end{cases} \qquad \begin{cases} x_2 \equiv +x_p \pmod p, \\ x_2 \equiv -x_q \pmod q, \end{cases}$$

$$\begin{cases} x_3 \equiv -x_p \pmod p, \\ x_3 \equiv +x_q \pmod q, \end{cases} \qquad \begin{cases} x_4 \equiv -x_p \pmod p, \\ x_4 \equiv -x_q \pmod q. \end{cases}$$

Найденные решения x_1, x_2, x_3, x_4 будут решениями сравнения (4.13).

Пример 29. Решить сравнение $x^2 \equiv 36 \pmod{77}$. Разложим 77 на простые множители $77 = 7 \cdot 11$. Решим два квадратичных сравнения по простому модулю: $x_p^2 \equiv 36 \equiv 1 \pmod 7$ и $x_q^2 \equiv 36 \equiv 3 \pmod{11}$ и получим $x_p \equiv \pm 1 \pmod 7$, а $x_q \equiv \pm 5 \pmod{11}$. Составим четыре системы:

$$\begin{cases} x_1 \equiv +1 \pmod 7, \\ x_1 \equiv +5 \pmod{11}, \end{cases} \qquad \begin{cases} x_2 \equiv +1 \pmod 7, \\ x_2 \equiv -5 \pmod{11}, \end{cases}$$

$$\begin{cases} x_3 \equiv -1 \pmod 7, \\ x_3 \equiv +5 \pmod{11}, \end{cases} \qquad \begin{cases} x_4 \equiv -1 \pmod 7, \\ x_4 \equiv -5 \pmod{11}. \end{cases}$$

Решениями этих систем будут ± 6, $\pm 27 \pmod{77}$. Заметим, что сложность решения квадратичного сравнения по составному модулю имеет ту же сложность, что и разложение модуля на множители.

4.13. Символ Лежандра

При изучении сравнений второй степени удобно пользоваться символом Лежандра. Символ Лежандра для числа a по простому модулю $p > 2$ принято записывать в виде $\left(\frac{a}{p}\right)$ или в виде $L(a, p)$, причем этот символ определяется следующим образом.

Определение 21. Пусть p — простое число, $p > 2$. *Символ Лежандра* определяется следующим образом:

$$L(a,p) = \begin{cases} 0, & \text{если } a \equiv 0 \pmod p; \\ 1, & \text{если } a - \text{квадратичный вычет по модулю } p; \\ -1, & \text{если } a - \text{квадратичный невычет по модулю } p. \end{cases}$$

Пример 30. Пусть $a = 3$, а $p = 11$, тогда $L(3, 11) = 1$, так как сравнение $x^2 \equiv 3 \pmod{11}$ имеет два решения: $x = \pm 5 \pmod{11}$, *а* $L(2, 5) = -1$, так как сравнение $x^2 \equiv 2 \pmod 5$ не имеет решений.

Приведем основные свойства символа Лежандра [6].

1. Если $a \equiv b \pmod p$ то $L(a, p) = L(b, p)$.
2. $L(a_1 a_2 ... a_k, p) = L(a_1, p)L(a_2, p)...L(a_k, p)$.
3. $L(a^2, p) = 1$, если $p \nmid a$.
4. $L(1, p) = 1$.
5. $L(a, p) = a^{(p-1)/2} \pmod p$ (критерий Эйлера).
6. $L(-1, p) = (-1)^{(p-1)/2}$.
7. $L(2, p) = (-1)^{\frac{p^2-1}{8}}$.

8. Если p, q два различных нечетных простых числа, тогда $L(q,p) = (-1)^{\frac{(p-1)(q-1)}{4}} L(p,q)$ (закон взаимности).

Предложение 9. *Если* $a \equiv b \pmod p$, *то* $L(a,p) = L(b,p)$.

Доказательство. Если $L(a,p) = 1$, т. е. a — квадратичный вычет по модулю p, то и любое $b \in [a]$ тоже будет квадратичным вычетом по этому модулю, и $L(b,p) = 1$. Если $L(a,p) = -1$, то и весь класс $[a]$ состоит из квадратичных невычетов по модулю p, т. е. при $a \equiv b \pmod p$, $L(b,p) = -1$. □

Предложение 10. $L(ab,p) = L(a,p)L(b,p)$.

Доказательство. Воспользуемся критерием Эйлера:

$$L(ab,p) \equiv (ab)^{\frac{p-1}{2}} \equiv a^{\frac{p-1}{2}} b^{\frac{p-1}{2}} \equiv L(a,p)L(b,p) \pmod p.$$

С учетом множества допустимых значений это сравнение можно превратить в равенство. □

Сформулируем алгоритм вычисления символа Лежандра:

1. Если число a отрицательное, то выделяем множитель $L(-1,p)$.
2. Заменяем число a на остаток от деления числа a на p.
3. Раскладываем число a в произведение степеней простых сомножителей $a = a_1^{i_1} a_2^{i_2} ... a_k^{i_k}$. Если число на простые множители не разлагается, то переходим на шаг 7.
4. Переходим к разложению $L(a,p) = L(a_1,p)^{i_1} L(a_2,p)^{i_2} ... L(a_k,p)^{i_k}$
5. Отбрасываем множители с четным значением показателя $i_j, j = \overline{1,k}$.
6. Вычисляем символ Лежандра $L(a_j,p)^{i_j}, a_j = 2$.
7. Если все символы Лежандра в выражении $L(a_1,p)^{i_1} L(a_2,p)^{i_2} ... L(a_k,p)^{i_k}$ известны, то вычисления $L(a,p)$ завершаются. В противном случае для множителей $L(a_j,p)^{i_j}$, у которых $a_j \neq 2$, применяем закон взаимности $L(q,a) = (-1)^{\frac{(a-1)(q-1)}{4}} L(a,q)$.
8. Переходим к шагу 1.

Рассмотрим пример вычисления символа Лежандра.

Пример 31. Вычислить $L(68,113)$. Здесь $a = 68$, а $p = 113$ есть простое число.

$$L(68,113) = L(2^2 \cdot 17, 113) \overset{2}{=} L(2^2,113)L(17,113) \overset{2}{=}$$

$$\overset{2}{=} L(2,113)^2 L(17,113) \overset{3}{=} L(17,113) = L(17,113) \overset{8}{=}$$

$$\overset{8}{=} (-1)^{8 \cdot 56} L(113,17) = L(113,17) \overset{1}{=} L(11,17) =$$

$$= (-1)^{5\cdot 8} L(17,11) = L(17,11) \overset{1}{=} L(6,11) = L(2\cdot 3,11) \overset{2}{=}$$

$$\overset{2}{=} L(2,11)L(3,11) \overset{7}{=} (-1)^{(11^2-1)/8} L(3,11) = -L(3,11) =$$

$$= -L(3,11) \overset{8}{=} -(-1)^{5\cdot 1} L(11,3) = L(11,3) =$$

$$= -L(3,11) \overset{8}{=} -(-1)^{5\cdot 1} L(11,3) = L(11,3) =$$

$$= L(11,3) \overset{1}{=} L(2,3) \overset{7}{=} (-1)^{(3^2-1)/8} = -1.$$

Итак, значение символа Лежандра $L(68,13) = -1$ позволяет сделать вывод, что сравнение $x^2 \equiv 68 \pmod{113}$ не имеет решений.

4.14. Возведение в степень и логарифмы

Рассмотрим сравнение

$$a^x \equiv y \pmod{n}, \tag{4.14}$$

где a, $y \in \mathbb{Z}$, $n \in \mathbb{N}$ известны и требуется найти $x \in \mathbb{Z}$. Если для возведения в степень известен эффективный алгоритм 4.3, то для решения сравнения (4.14) эффективный алгоритм неизвестен. Первый алгоритм, который возможен — алгоритм полного перебора. Поскольку по теореме Эйлера $a^{\varphi(n)} \equiv 1 \pmod{n}$, то имеет смысл перебрать значения x только в диапазоне $0, 1, .., \varphi(n) - 1$. Время работы $\mathcal{O}(nlog(n))$.

Алгоритм 4.4 Алгоритм перебора для вычисления логарифма

```
1: procedure LOG(a,y, n)                    ▷ Алгоритм вычисляет Log_a(y)
2:     x ← 1
3:     while 1 do
4:         if FastPow(a, x, n) = y then
5:             return x
6:         end if
7:         x ← x + 1
8:     end while
9: end procedure
```

Рассмотрим две группы $\mathbb{Z}_8^*$ (табл. 4.8) и $\mathbb{Z}_7^*$ (табл. 4.9). Поскольку $\varphi(8) = 4$, а $\varphi(7) = 6$, то элементы в таблицах возведены до степени $\varphi(n)$. Квадратными скобками выделены первый единичный элемент. Несмотря на то, что теорема Эйлера нам гарантирует, что последний столбец

должен быть заполнен единичными элементами, многие элементы обращаются в единицу ранее. Причем поскольку номер x первого обращения в единицу — это по определению порядок элемента, то это обращение возможно только при $x|\varphi(n)$.

Определение 22. Элемент $\mathbb{Z}_n^*$, порядок которого совпадает с порядком группы, называют *первообразным корнем.*

Теорема 16. *Первообразные корни существуют только по модулям вида* $n = 2,\ 4,\ p^a,\ 2p^a$, *где* $p > 2$ *— простое число.*

Теорема 17. *Если по модулю* n *существует первообразный корень, то всего существует* $\varphi(\varphi(n))$ *различных первообразных корней по модулю* n.

Упражнение 27. Докажите, что если группа $\mathbb{Z}_n^*$ имеет первообразные корни, то она является циклической группой.

Если a является первообразным корнем n, то степени $a,\ a^2, ...,\ a^{\varphi(n)}$ оказываются различными по модулю n и взаимно простыми с n. В частности, для простого p, если a является первообразным корнем, то $a,\ a^2, ...,\ a^{p-1}$ оказываются различными по модулю p.

Таблица 4.8
Возведение в степень в группе $\mathbb{Z}_8^*$

$a^x \bmod 8$	$x=1$	$x=2$	$x=3$	$x=4$
$a=1$	$\boxed{1}$	1	1	1
$a=3$	3	$\boxed{1}$	3	1
$a=5$	5	$\boxed{1}$	5	1
$a=7$	7	$\boxed{1}$	7	1

А поскольку в группе $\mathbb{Z}_p^*$ всего $p-1$ элементов, то элемент a — образующий элемент этой циклической группы. Следовательно, для каждого $b = \overline{1, p-1}$ можно найти такой индекс i, что $b \equiv a^i \pmod p$, $0 \le i \le p-2$. Значение i называется индексом числа b по модулю p при основании a. Будем его обозначать как $\mathrm{ind}_{a,p}(b)$. Поскольку по теореме Эйлера $a^{\varphi(n)} \equiv 1 \pmod n$, то $\mathrm{ind}_{a,p}(b)$ образуют класс вычетов по модулю $\varphi(n)$. Мы сразу можем заметить, что $\mathrm{ind}_{a,p}(1) = 0$, т.к. $a^0 \bmod p = a^{\varphi(p)} \bmod p = 1$. А $\mathrm{ind}_{a,p}(a) = 1$, т.к. $a^1 \bmod p = a$. Тогда по определению выполняются следующие три сравнения:

$$\begin{aligned} x &\equiv a^{\mathrm{ind}_{a,p}(x)} \pmod p; \\ y &\equiv a^{\mathrm{ind}_{a,p}(y)} \pmod p; \\ xy &\equiv a^{\mathrm{ind}_{a,p}(xy)} \pmod p. \end{aligned} \tag{4.15}$$

С учетом (4.15) можно записать, что

$$a^{\mathrm{ind}_{a,p}(xy)} \pmod p \equiv xy \equiv$$
$$\equiv \left(a^{\mathrm{ind}_{a,p}(x)} \pmod p\right)\left(a^{\mathrm{ind}_{a,p}(y)} \pmod p\right) \equiv a^{\mathrm{ind}_{a,p}(x)+\mathrm{ind}_{a,p}(y)} \pmod p.$$

А поскольку a первообразный корень, то

$$\mathrm{ind}_{a,p}(xy) \equiv \mathrm{ind}_{a,p}(x) + \mathrm{ind}_{a,p}(y) \pmod{\varphi(p)}. \tag{4.16}$$

Обобщая (4.16) можно получить сравнение $\mathrm{ind}_{a,p}(x^r) \equiv r \cdot \mathrm{ind}_{a,p}(x) \pmod{\varphi(p)}$. Поскольку свойства индекса по модулю (табл. 4.11) похожи на свойства логарифма, то его часто называют дискретным логарифмом.

Вернемся к решению сравнения (4.14). С использованием дискретного логарифма мы знаем, что если a - первообразный корень, то решение существуют. Проще всего найти его с помощью таблицы логарифмов 4.10. Например решение сравнения $4 \equiv 3^x \pmod 7$ по таблице 4.10 будет равно $x \equiv 4 \pmod 6$. Сравнение $6 \equiv 5^x \pmod 7$ также легко решается $x \equiv 3 \pmod 6$. Однако, составление таких таблиц процедура дорогостоящая: нужно много времени и памяти.

Таблица 4.9

Возведение в степень в группе $\mathbb{Z}_7^*$

$a^x \bmod 7$	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$
$a=1$	[1]	1	1	1	1	1
$a=2$	2	4	[1]	1	4	1
$a=3$	3	2	6	4	5	[1]
$a=4$	4	2	[1]	4	2	1
$a=5$	5	4	6	2	3	[1]
$a=6$	6	[1]	6	1	6	1

Таблицы и свойства дискретных логарифмов не могут применяться для решения сравнения (4.14), когда n является очень большим. Для решения этой проблемы было разработано несколько алгоритмов, в которых используется основная идея дискретных логарифмов. Хотя все эти алгоритмы более эффективны, чем алгоритмы полного перебора, но ни один из них не имеет полиномиальной сложности. Большинство этих алгоритмов имеет такой же уровень сложности, как задача разложения на множители.

Таблица 4.10

Дискретный логарифм для группы $\mathbb{Z}_7^*$

$\mathrm{ind}_{a,7}(b)$	$b=1$	$b=2$	$b=3$	$b=4$	$b=5$	$b=6$
$a=3$	0	2	1	4	5	3
$a=5$	0	4	5	2	1	3

Алгоритм больших и малых шагов для вычисления дискретного логарифма 4.5 позволяет находить дискретный логарифм с помощью двух специальных таблиц. Докажем, что алгоритм корректен.

Предложение 11. *Алгоритм 4.5 работает корректно.*

Алгоритм 4.5 Алгоритм больших и малых шагов для вычисления дискретного логарифма

1: **procedure** DLOG(a,b,p) ▷ Алгоритм вычисляет $\mathrm{ind}_a, p(b)$, т.е. решает сравнение $b \equiv a^x \pmod{p}$
2: $H \leftarrow \lfloor \sqrt{p} \rfloor + 1$
3: $c \leftarrow a^H \bmod p$
4: Составить таблицу значений $c^u \bmod p$ для $1 \leq u \leq H$ и отсортировать её.
5: Составить таблицу значений $b \cdot a^v \bmod p$ для $0 \leq v \leq H$ и отсортировать её.
6: Найти общие элементы в таблицах. Для них $c^u \equiv b \cdot a^v \pmod{p}$, откуда $a^{Hu-v} \equiv b \pmod{p}$.
7: **return** $Hu - v$
8: **end procedure**

Таблица 4.11

Сравнение дискретных и обычных логарифмов

Обычный логарифм	Дискретный логарифм
$\log_a(1) = 0$	$\mathrm{ind}_{a,p}(1) \equiv 0 \pmod{\varphi(p)}$
$\log_a(a) = 1$	$\mathrm{ind}_{a,p}(a) \equiv 1 \pmod{\varphi(p)}$
$\log_a(xy) = \log_a(x) + \log_a(y)$	$\mathrm{ind}_{a,p}(xy) \equiv \mathrm{ind}_{a,p}(x) + \mathrm{ind}_{a,p}(y) \pmod{\varphi(p)}$
$\log_a(x^r) = r \cdot \log_a(x)$	$\mathrm{ind}_{a,p}(x^r) \equiv r \cdot \mathrm{ind}_{a,p}(x) \pmod{\varphi(p)}$

Доказательство. Любое целое число x, $0 \leq x \leq p-2$ можно представить как $x \equiv Hu - v \pmod{p-1}$, где $1 \leq u \leq H$ и $0 \leq v \leq H$. Правая часть сравнения порождает набор чисел $0, 1, ..., H^2$, а поскольку $H^2 > p$, то правая часть порождает все числа $0, 1, ..., p-2$. Следовательно, алгоритм корректен. □

Время работы алгоритма определяется временем сортировки двух списков и составляет $\mathcal{O}(\sqrt{p}\log(p))$. Другой алгоритм для решения задачи дискретного логарифмирования был предложен С. Полигом и М. Хеллманом в 1978 г.

Алгоритм Полига-Хеллмана применим, когда известно разложение числа $p-1$ на простые множители $p - 1 = \prod_{i=1}^{k} q_i^{\alpha_i}$. Тогда решение сравнения

$$a^x \equiv b \pmod{p} \tag{4.17}$$

Алгоритм 4.6 Алгоритм Полига-Хеллмана для вычисления дискретного логарифма

1: **procedure** POHLIG–HELLMAN(a,b,p,q_i,α_i) ▷ Алгоритм вычисляет $\text{ind}_a, p(b)$ при известном разложении на множители $p-1=\prod_{i=1}^{k} q_i^{\alpha_i}$.

2: Составить таблицу значений $\{r_{i,j}\}$, где $r_{i,j}=a^{j\cdot\frac{p-1}{q_i}}$, $i=\overline{1,k}$, $j=\overline{0,q_i-1}$.

3: **for all** $i=\overline{1,k}$ **do**

4: С помощью таблицы $\{r_{i,j}\}$ решить сравнение $a^{x_0\cdot\frac{p-1}{q_i}}\equiv b^{\frac{p-1}{q_i}} \pmod p$ и найти x_0

5: **for all** $j=\overline{1,\alpha_i-1}$ **do**

6: Аналогично решить сравнение $a^{x_j\cdot\frac{p-1}{q_i}}\equiv (ba^{-x_0-x_1q_i-\ldots-x_{j-1}q_i^{j-1}})^{\frac{p-1}{q_i^{j+1}}} \pmod p$ и найти x_j

7: **end for**

8: $y_i\leftarrow x_0+x_1q_i+\ldots+x_{\alpha_i-1}q_i^{\alpha_i-1}$

9: **end for**

10: С помощью китайской теоремы об остатках решить систему сравнений $\begin{cases} x\equiv y_1 \pmod{q_1^{\alpha_1}} \\ x\equiv y_2 \pmod{q_2^{\alpha_2}} \\ \ldots \\ x\equiv y_k \pmod{q_k^{\alpha_k}} \end{cases}$

11: **return** x

12: **end procedure**

сводится к нахождению $y_i=\text{ind}_{a,p}(b) \bmod q_i^{\alpha_i}$, $i=\overline{1,k}$ и решению системы сравнений

$$\begin{cases} x\equiv y_1 \pmod{q_1^{\alpha_1}}; \\ x\equiv y_2 \pmod{q_2^{\alpha_2}}; \\ \ldots \\ x\equiv y_k \pmod{q_k^{\alpha_k}}. \end{cases}$$

Алгоритм 4.6 реализует описанный алгоритм. Во второй строке организуется вспомогательная таблица логарифмов для быстрого вычисления небольших значений логарифмов. Также возможен подход, когда для небольших значений логарифмов используется алгоритм больших и малых шагов алг. 4.5. Затем в цикле 3–9 организуется поиск значений y_i. Для этого представим y_i разложенный по степеням q_i

$$y_i=\text{ind}_{a,p}(b) \bmod q_i^{\alpha_i}=x \bmod q_i^{\alpha_i}=x_0+x_1q_i+\ldots+x_{\alpha-1}q_i^{\alpha-1}, \quad (4.18)$$

где $0\le x_i\le q_i-1$.

Возведём левую и правую части сравнения (4.18) в степень $(p-1)/q_i$:

$$a^{x\cdot\frac{p-1}{q_i}} \equiv b^{\frac{p-1}{q_i}} \pmod{p}.$$

Подставим (4.18) и преобразуем сравнение:

$$a^{x_0\cdot\frac{p-1}{q_i}+x_1\cdot(p-1)+x_2\cdot q_i\cdot(p-1)+\cdots+x_{\alpha_i-1}\cdot q_i^{\alpha_i-2}\cdot(p-1)} \equiv b^{\frac{p-1}{q_i}} \pmod{p}.$$

Распишем степени

$$a^{x_0\cdot\frac{p-1}{q_i}}\cdot a^{x_1\cdot(p-1)}\cdot a^{x_2\cdot q_i\cdot(p-1)}\cdot\ldots\cdot a^{x_{\alpha_i-1}\cdot q_i^{\alpha_i-2}\cdot(p-1)} \equiv b^{\frac{p-1}{q_i}} \pmod{p}.$$

Т.к. a — первообразный корень, следовательно, верны сравнения вида: $a^{m\cdot(p-1)} \equiv 1 \pmod{p}$, $\forall m \in \{0,1,\ldots,p-1\}$. Получаем

$$a^{x_0\cdot\frac{p-1}{q_i}}\cdot 1\cdot 1\cdot\ldots\cdot 1 \equiv b^{\frac{p-1}{q_i}} \pmod{p},$$

$$a^{x_0\cdot\frac{p-1}{q_i}} \equiv b^{\frac{p-1}{q_i}} \pmod{p}. \tag{4.19}$$

Решая (4.19), можно найти x_0 разложения (4.18) (строка 4 в алгоритме 4.6).

Поскольку x_0 теперь известно, то возведем левую и правую часть сравнения (4.17) в степень $(p-1)/q_i^2$ и получим

$$a^{x_0\cdot\frac{p-1}{q_i^2}+x_1\cdot\frac{p-1}{q}+x_2\cdot(p-1)+\cdots+x_{\alpha_i-1}\cdot q_i^{\alpha_i-3}\cdot(p-1)} \equiv b^{\frac{p-1}{q_i^2}} \pmod{p}.$$

Перенесем все известные величины вправо, для этого домножим на $a^{-x_0\frac{p-1}{q_i^2}}$

$$a^{x_1\cdot\frac{p-1}{q}+x_2\cdot(p-1)+\cdots+x_{\alpha_i-1}\cdot q_i^{\alpha_i-3}\cdot(p-1)} \equiv \left(ba^{-x_0}\right)^{\frac{p-1}{q_i^2}} \pmod{p}.$$

Аналогично получаем

$$a^{x_1\cdot\frac{p-1}{q}} \equiv \left(ba^{-x_0}\right)^{\frac{p-1}{q_i^2}} \pmod{p}. \tag{4.20}$$

Сравнение (4.20) решаем аналогично (4.19). В результате в строке 8 формируется значение y_i, а в строке 10 находится ответ. Время работы алгоритма $\mathcal{O}\left(\sum\limits_{i=1}^{k}\alpha_i(\log p+\sqrt{q}_i)\right)$.

Пример 32. Решить сравнение $2^x \equiv 28 \pmod{37}$.

Находим разложение $\varphi(37) = 37-1 = 36 = 2^2\cdot 3^2$.

Получаем $q_1 = 2, \alpha_1 = 2, q_2 = 3, \alpha_2 = 2$.

Составляем таблицу r_{ij} :

$r_{10} \equiv 2^{0\cdot\frac{37-1}{2}} \equiv 1 \pmod{37}$;

$r_{11} \equiv 2^{1\cdot\frac{37-1}{2}} \equiv 2^{18} \equiv -1 \pmod{37}$;

$r_{20} \equiv 2^{0 \cdot \frac{37-1}{3}} \equiv 1 \pmod{37}$;
$r_{21} \equiv 2^{1 \cdot \frac{37-1}{3}} \equiv 2^{12} \equiv 26 \pmod{37}$;
$r_{22} \equiv 2^{2 \cdot \frac{37-1}{3}} \equiv 2^{24} \equiv 10 \pmod{37}$.

Рассматриваем $q_1 = 2$. Для x верно: $x \equiv x_0 + x_1 q_1 \pmod{q_1^{\alpha_i}} \equiv x_0 + x_1 \cdot 2 \pmod{2^2}$. Находим x_0 из сравнения:

$$a^{x_0 \cdot \frac{p-1}{q_1}} \equiv b^{\frac{p-1}{q_1}} \pmod{p} \Rightarrow 2^{x_0 \cdot \frac{37-1}{2}} \equiv 28^{\frac{37-1}{2}} \equiv 28^{18} \equiv 1 \pmod{37}.$$

Из таблицы находим, что при $x_0 = 0$ верно выше полученное сравнение. Находим x_1 из сравнения:

$$a^{x_1 \cdot \frac{p-1}{q_i}} \equiv \left(b \cdot a^{-x_0}\right)^{\frac{p-1}{q_i^2}} \Rightarrow 2^{x_1 \cdot \frac{37-1}{2}} \equiv \left(28 \cdot 2^{-0}\right)^{\frac{37-1}{4}} \equiv 28^9 \equiv -1 \pmod{37}.$$

Из таблицы получаем, что при $x_1 = 1$ верно выше полученное сравнение. Находим $x \equiv 0 + 1 \cdot 2 \equiv 2 \pmod{4}$. Теперь рассматриваем $q_2 = 3$. Для x верно $x \equiv x_0 + x_1 \cdot 3 \pmod{3^2}$. По аналогии находим x_0 и x_1:

$$2^{x_0 \cdot \frac{37-1}{3}} \equiv 28^{\frac{37-1}{3}} \equiv 28^{12} \equiv 26 \pmod{37} \Rightarrow x_0 = 1,$$

$$2^{x_1 \cdot \frac{37-1}{3}} \equiv \left(28 \cdot 2^{-1}\right)^{\frac{37-1}{3^2}} \equiv 14^4 \equiv 10 \pmod{37} \Rightarrow x_1 = 2.$$

Получаем $x \equiv 1 + 2 \cdot 3 \equiv 7 \pmod{9}$. Получаем систему

$$\begin{cases} x \equiv 2 \pmod{4}; \\ x \equiv 7 \pmod{9}. \end{cases}$$

Можно воспользоваться китайской теоремой об остатках, но мы продемонстрируем еще один подход. Первое сравнение преобразуем в равенство, которое подставляем во второе сравнение:

$$x = 2 + 4 \cdot t \Rightarrow 2 + 4 \cdot t \equiv 7 \pmod{9} \Rightarrow 4 \cdot t \equiv 5 \pmod{9} \Rightarrow$$

$$\Rightarrow t \equiv 5 \cdot (4)^{-1} \equiv 5 \cdot (-2) \equiv -10 \equiv 8 \pmod{9}$$

Подставляем найденное t и получаем искомое x:

$$x \equiv 2 + 4 \cdot 8 \equiv 34 \pmod{36}.$$

Ответ $x = 34 + 36k$, $k \in \mathbb{Z}$.

Упражнение 28. Найти порядок группы $G = \langle \mathbb{Z}_{21}^*, * \rangle$. Перечислите все элементы и укажите первообразные корни.

Упражнение 29. Найти порядок группы $G = \langle \mathbb{Z}_{10}^*, * \rangle$. Перечислите все элементы и укажите первообразные корни

Упражнение 30. Найти порядок группы $G = \langle \mathbb{Z}_8^*, * \rangle$. Перечислите все элементы и укажите первообразные корни

Упражнение 31. Сколько первообразных корней в группе $G = \langle \mathbb{Z}_{10}^*, * \rangle$?

5. ПРОСТЫЕ ЧИСЛА

Как мы видели ранее, для построения полей $\mathbb{Z}_p$ требуется находить большие простые числа. На вопрос существует ли наибольшее простое число дает ответ следующая теорема.

Теорема 18. *Множество простых чисел бесконечно.*

Доказательство. Докажем от противного. Пусть множество простых чисел конечно, тогда простые числа можно записать как p_1, p_2, ..., p_n. Рассмотрим число $P = p_1 p_2 ... p_n + 1$. Это число не делится ни на одно простое число (простые числа по предположению p_1, p_2, ..., p_n), но имеет как минимум два различных делителя, следовательно, наше предположение неверно. □

Пример 33. Перечислите простые числа, меньшие, чем 10.

Есть четыре простых числа меньше чем 10: 2, 3, 5 и 7. Несмотря на то, что простых чисел бесконечно много, можно оценить их количество в конкретном диапазоне.

Определение 23. *Функция распределения простых чисел* $\pi(x)$ — это функция, численно равна количеству простых чисел, меньших либо равных действительному числу x.

Например, $\pi(1) = 0$, $\pi(1) = 0$, $\pi(2) = 1$, $\pi(3) = 2$, $\pi(10) = 4$, $\pi(20) = 8$, $\pi(50) = 8$, $\pi(100) = 25$. Для функции $\pi(x)$ справедлива теорема о распределении простых чисел, которая утверждает, что

$$\lim_{x \to +\infty} \frac{\pi(x)}{x/\ln(x)} = 1.$$

Кроме того, для оценки количества чисел в диапазоне можно использовать неравенство

$$\frac{x}{\ln(x)} < \pi(x) < 1,26 \frac{x}{\ln(x)},$$

при $x \geq 17$.

Для поиска простых чисел, меньше чем n, Эратосфен (200 г. до н. э.) изобрел метод, называемый решетом Эратосфена. Это метод заключается в следующем: выписываются все числа от 2 до n, начиная по порядку, вычеркиваются все числа кратные первому не вычеркнутому числу, затем вычеркиваются все числа кратные второму не вычеркнутому числу и т.д. Пример решета Эратосфена для $n = 100$ приведен в табл. 5.1.

В таблице 5.1 были выписаны все числа от 2 до n. Затем были вычеркнуты все числа кратные двум. Эти числа перечеркнуты и помечены индексом 2. После этого было выбрано следующее не вычеркнутое числа — три и были вычеркнуты все числа кратные трем, числа которые уже были вычеркнуты не рассматривались. Следующее число было выбрано 5, поскольку 4 было вычеркнуто при рассмотрении двойки. Такая последовательность действий продолжалась для всех чисел от 2 до 100. В результате остались только простые числа от 2 до 100, они выделены жирным шрифтом.

Таблица 5.1

Пример решета Эратосфена

	2	**3**	~~4_2~~	**5**	~~6_2~~	**7**	~~8_2~~	~~9_3~~	~~10_2~~
11	~~12_2~~	**13**	~~14_2~~	~~15_3~~	~~16_2~~	**17**	~~18_2~~	**19**	~~20_2~~
~~21_3~~	~~22_2~~	**23**	~~24_2~~	~~25_5~~	~~26_2~~	~~27_3~~	~~28_2~~	**29**	~~30_2~~
31	~~32_2~~	~~33_3~~	~~34_2~~	~~35_5~~	~~36_2~~	**37**	~~38_2~~	~~39_3~~	~~40_2~~
41	~~42_2~~	**43**	~~44_2~~	~~45_3~~	~~46_2~~	**47**	~~48_2~~	~~49_7~~	~~50_2~~
~~51_3~~	~~52_2~~	**53**	~~54_2~~	~~55_5~~	~~56_2~~	~~57_3~~	~~58_2~~	**59**	~~60_2~~
61	~~62_2~~	~~63_3~~	~~64_2~~	~~65_5~~	~~66_2~~	**67**	~~68_2~~	~~69_3~~	~~70_2~~
71	~~72_2~~	**73**	~~74_2~~	~~75_3~~	~~76_2~~	~~77_7~~	~~78_2~~	**79**	~~80_2~~
~~81_3~~	~~82_2~~	**83**	~~84_2~~	~~85_5~~	~~86_2~~	~~87_3~~	~~88_2~~	**89**	~~90_2~~
~~91_7~~	~~92_2~~	~~93_3~~	~~94_2~~	~~95_5~~	~~96_2~~	**97**	~~98_2~~	~~99_3~~	~~100_2~~

Алгоритм 5.1 Решето Эратосфена

```
1:  procedure SieveOfEratosthenes(n)        ▷ Алгоритм возвращает список простых чисел от 2 до n
2:      for all i = 2, n do
3:          A[i] ← 1
4:      end for
5:      for i = 2, ⌊√n⌋ do
6:          if A[i] = 1 then
7:              for all j = i^2, i^2 + 1 · i, i^2 + 2 · i, ..., i^2 + ⌊n/i − i⌋ · i do
8:                  A[j] ← 0
9:              end for
10:         end if
11:     end for
12:     return {i | A[i] = 1, i = 2, n}
13: end procedure
```

Псевдокод алгоритма 5.1 в строках 2–4 инициализирует массив меток $A[i]$. В строках 5–11 выполняется проход по всем не вычеркнутым числам и вычеркивание в строках 7–9. В строке 12 алгоритм возвращает множество не вычеркнутых чисел.

Упражнение 32. Докажите, что алгоритм работает корректно, если i в 5 строке алгоритма 5.1 меняется до $\lfloor\sqrt{n}\rfloor$.

Упражнение 33. Докажите, что алгоритм работает корректно, если j в 7 строке алгоритма 5.1 меняется от i^2.

Упражнение 34. Докажите, что алгоритм работает корректно, если j в 7 строке алгоритма 5.1 меняется до $i^2 + \left\lfloor \frac{n}{i} - i \right\rfloor \cdot i$.

Упражнение 35. Продемонстрируйте работу алгоритма 5.1 при $n = 200$.

Упражнение 36. Какова оценка времени работы алгоритма 5.1 ?

5.1. Генерация простых чисел

Поскольку для криптографии очень важно уметь находить простые числа, то вопрос существования формулы для генерации простых чисел является актуальным. Рассмотрим два класса простых чисел: числа Мерсенна и Ферма.

5.1.1. Числа Мерсенна

Числами Мерсенна называют числа, вычисленные по формуле

$$M_p = 2^p - 1,$$

где p — простое число. Простые числа такого вида называют простыми числами Мерсенна. Первоначально предполагалось, что если p в приведенной выше формуле — простое число, то, как предполагали, M_p должно быть простым числом. Однако, это верно не для всех p. Рассмотрим первые числа Мерсенна.

$M_2 = 2^2 - 1 = 3$ — простое;
$M_3 = 2^3 - 1 = 8$ — простое;
$M_5 = 2^5 - 1 = 31$ — простое;
$M_7 = 2^7 - 1 = 127$ — простое;
$M_{11} = 2^{11} - 1 = 2047$ — составное $2047 = 23 \cdot 89$;
$M_{13} = 2^{13} - 1 = 8191$ — простое;
$M_{17} = 2^{17} - 1 = 131071$ — простое.

Оказалось, что M_{11} — не простое число. На 01.12.2014 было найдено 48 простых числа Мерсенна. Для чисел Мерсенна существует эффективный тест проверки простоты — тест Люка-Лемера. Благодаря этому тесту самыми большими известными простыми числами чаще всего были простые числа Мерсенна, причём даже до появления компьютеров. Тест Люка-Лемера базируется на следующей теореме.

Теорема 19 (тест Люка-Лемера)**.** *Пусть q — простое число, $q > 2$, $n = 2^q - 1$. Рассмотрим последовательность L_0, L_1, ..., L_{q-2}, определяемую соотношениями $L_0 = 4$, $L_{j+1} = \left(L_j^2 - 2\right) \bmod n$, $j = \overline{0, q-1}$. Число n простое тогда и только тогда, когда $L_{q-2} \equiv 0 \pmod{n}$.*

Упражнение 37. Верно ли утверждение, что если $2^p - 1$ — простое, то p — также простое?

5.1.2. Числа Ферма

Ферма предложил следующую формулу

$$F_n = 2^{2^n} + 1,$$

которая теперь называется формулой Ферма, и проверил числа F_0, F_1, ..., F_4, но оказалось, что уже F_5 — не простое число. Рассмотрим первые числа Ферма.

$F_0 = 2^{2^0} + 1 = 3$ — простое;
$F_1 = 2^{2^1} + 1 = 5$ — простое;
$F_2 = 2^{2^2} + 1 = 17$ — простое;
$F_3 = 2^{2^3} + 1 = 257$ — простое;
$F_4 = 2^{2^4} + 1 = 65537$ — простое;
$F_5 = 2^{2^5} + 1 = 4294967297 = 641 \cdot 6700417$ — составное.

Было доказано, что числа F_5, ..., F_{32} — составные числа. Для чисел Ферма эффективным является тест Пепина.

Теорема 20 (Тест Пепина)**.** *Число Ферма F_p является простым тогда и только тогда, когда* $3^{(F_p-1)/2} \equiv -1 \pmod{F_p}$.

5.2. Испытание простоты чисел

Поскольку формулы получения простых чисел, подобно формулам Ферма или Мерсенна, не гарантируют, что полученные числа — простые, то как можно сгенерировать большие простые числа, необходимые для криптографии? Можно выбрать случайно большое число и провести испытание, чтобы убедиться, что оно — простое. Рассмотрим некоторые из таких тестов простоты далее.

5.2.1. Тест пробных делений

Возьмем некоторое натуральное число n. Воспользуемся определением и проверим имеет ли число n делители от 2 до $\sqrt{n}$, на которые n делится без остатка. Мы рассматриваем делители только до $\sqrt{n}$, т.к. минимальный делитель (отличный от единицы) составного числа должен быть не больше $\sqrt{n}$. Если минимальный делитель $p > \sqrt{n}$, то и $n/p > \sqrt{n}$, но тогда, перемножая эти неравенства, получим противоречие $n = p \cdot n/p > \sqrt{n} \cdot \sqrt{n} = n$.

Такой подход представлен в алгоритме 5.2. Очевидно, что перебор можно сократить, если проверять не все числа, а только простые.

Пример 34. Является ли число 97 простым?

Алгоритм 5.2 Проверка на простоту числа

```
1: procedure CheckPrime(n)  ▷ Алгоритм проверки простоты числа n, n > 1
2:     for i = 2, ⌊√n⌋ do
3:         if n mod i = 0 then
4:             return NOT_PRIME
5:         end if
6:     end for
7:     return PRIME
8: end procedure
```

Вычислим $\left\lfloor \sqrt{97} \right\rfloor = 9$. Следовательно, требуется проверить остатки от деления 97 на 2, 3, 4, 5, 6, 7, 8, 9 или (с учетом простоты) 2, 3, 5 и 7. Проверка показывает, что 97 простое число.

Пример 35. Является ли число 301 простым?

Вычислим $\left\lfloor \sqrt{301} \right\rfloor = 17$. Мы должны проверить 2, 3, 5, 7, 11, 13 и 17. Числа 2, 3 и 5 не делят 301, но 7 — делит. Поэтому 301 — не простое число.

Упражнение 38. Докажите, что в алгоритме 5.2 в цикле можно перебирать только простые числа.

Упражнение 39. Каково время работы алгоритма 5.2?

Упражнение 40. Продемонстрируйте работу алгоритма 5.2 при $n = 191$.

Упражнение 41. Каково время работы алгоритма 5.2, если в цикле перебирать только простые числа?

5.2.2. Тест Ферма

Тест пробных делений работает очень долго и не подходит для проверки простоты больших чисел. Определим следующие положения как тест Ферма.

Если n — простое число, то $a^{n-1} \equiv 1 \pmod{n}$.

Если n — составное число, то возможно, что $a^{n-1} \equiv 1 \pmod{n}$.

В результате выполнения теста Ферма для выбранного основания a мы достоверно знаем, что число составное, если $a^{n-1} \not\equiv 1 \pmod{n}$. Если $a^{n-1} \equiv 1 \pmod{n}$, то либо n — простое, либо основание a выбрано неудачно. Таким образом, даже если для нескольких оснований тест Ферма показал, что число возможно простое по каждому из этих оснований,

Алгоритм 5.3 Тест Ферма на простоту числа

```
1: procedure FermatTest(n, a)     ▷ Алгоритм проводит тест Ферма для числа n по основанию a
2:     if a^{n-1} mod n = 1 then
3:         return PossiblePrime
4:     else
5:         return AuthenticallyNotPrime
6:     end if
7: end procedure
```

мы все равно не можем гарантировать, что число n простое. Вероятность того, что для составного числа n тест Ферма выдаст возможно простое, не превосходит $\varphi(n)/n$.

Пример 36. Проведите испытание Ферма для числа 561.

Используем в качестве основания число 2. Выполним возведение в степень $2^{561-1} \equiv 1 \pmod{561}$. Число прошло тест Ферма, но это — не простое число, потому что $561 = 3 \cdot 11 \cdot 17$.

Пример 37. Проведите испытание Ферма для числа 17.

Используем в качестве основания число 2. $2^{17-1} \equiv 1 \pmod{17}$. Число прошло тест Ферма, это — простое число.

Упражнение 42. Как выполнить тест Ферма для двух оснований? А для m оснований?

Упражнение 43. Составьте алгоритм проверки теста Ферма по нескольким основаниям.

Упражнение 44. Какова вероятность того, что составное число будет признано простым тестом Ферма, если используются k различных оснований для проверки числа, а вероятность успешного прохождения теста Ферма составным числом для одного основания равна ε?

Упражнение 45. Какова трудоемкость теста Ферма при проверке одного основания?

5.2.3. Тест Миллера—Рабина

Рассмотрим сравнение $x^2 \equiv 1 \pmod{p}$, где p — простое число. Возможность использования этого сравнения для тестирования простоты числа обосновывается следующим утверждением и обычно называется тестом квадратного корня.

Предложение 12. *Если p — простое число и $x^2 \equiv 1 \pmod{p}$, то $x \equiv \pm 1 \pmod{p}$.*

Доказательство. Рассмотрим сравнение

$$x^2 - 1 \equiv 0 \pmod{p},$$

перепишем это сравнение через равенство

$$x^2 - 1 = kp,\ k \in \mathbb{Z}.$$

Разложим на множители левую часть и получим

$$(x-1)(x+1) = kp. \tag{5.1}$$

В уравнении (5.1) правая часть делится на p, следовательно, левая часть также должна делиться на p. Число p — простое, следовательно, либо левая скобка должна делиться на p, либо правая. Т. е.

$$\left[\begin{array}{l} x - 1 = k_1 p,\ k_1 \in \mathbb{Z} \\ x + 1 = k_2 p,\ k_2 \in \mathbb{Z} \end{array}\right.$$

или возвращаясь к сравнениям $x \equiv \pm 1 \pmod{p}$. □

Таким образом, если n — простое число, то квадратный корень равен либо $+1$, либо -1. Если n — составное число, то квадратный корень равен $+1$ или -1, но могут быть и другие корни. Это называют испытанием простоты чисел квадратным корнем.

Пример 38. Рассмотрим пример $n = 7$. Возведем все элементы $\mathbb{Z}_7^*$ в квадрат и рассмотрим те, которые равны 1.

$1^2 \equiv 1 \pmod{7}$; $2^2 \equiv 4 \pmod{7}$; $3^2 \equiv 2 \pmod{7}$;
$4^2 \equiv 2 \pmod{7}$; $5^2 \equiv 4 \pmod{7}$; $6^2 \equiv 1 \pmod{7}$.

Только элементы 1 и 6 дали 1, следовательно, корень из 1 равен либо 1, либо -1.

Пример 39. Возведем все элементы $\mathbb{Z}_8^*$ в квадрат и рассмотрим те, которые равны 1.

$1^2 \equiv 1 \pmod{8}$; $2^2 \equiv 4 \pmod{8}$; $3^2 \equiv 1 \pmod{8}$; $4^2 \equiv 0 \pmod{8}$;
$5^2 \equiv 1 \pmod{8}$; $6^2 \equiv 4 \pmod{8}$; $7^2 \equiv 1 \pmod{8}$.

Как мы видим корень из единицы может быть равен 1, 3, 5, 7.

Тест квадратного корня эффективно применяется в тесте Миллера—Рабина, предложенном Г. Миллером и М. Рабином в 1980 г. Тест Миллера—Рабина определения простого числа есть комбинация тестов Ферма и квадратного корня. В этом тесте мы записываем $n - 1$ как произведение нечетного числа m и степени числа 2:

$$n - 1 = m \cdot 2^k.$$

В тесте Ферма по основанию a можно записать

$$a^{n-1} = a^{m \cdot 2^k} = (a^m)^{2^k} = (a^m)^{2 \cdot 2 \cdot \ldots \cdot 2}. \tag{5.2}$$

Вместо того чтобы вычислять $a^{n-1} \bmod n$ в один шаг, мы можем сделать это в $k+1$ шагов. Преимущество такого вычисления заключается именно в том, что испытание квадратным корнем может быть выполнено на каждом шаге. Если квадратный корень показывает сомнительные результаты, мы останавливаемся и объявляем n составным числом. На каждом шаге мы обеспечиваем, что прохождение теста Ферма и испытание квадратным корнем.

Алгоритм 5.4 реализует тест Миллера—Рабина. Перед запуском необходимо выбрать основание a. Во второй строке мы находим необходимые нам нечетное m и целое k. В третьей строке вычисляем $T = a^m$, где $m = (n-1)/2^k$. Для вычисления $a^{n-1} \bmod n$ по формуле (5.2) осталось T возвести k раз в квадрат. Но прежде чем продолжать вычисления проверим, что у нас получилось.

Если T равно $+1$ или -1, то это означает, что n на следующем шаге станет 1 и до конца цикла останется равным 1, т.е. $a^{n-1} \bmod n = 1$ и тест Ферма будет пройден. Кроме того, мы можем говорить, что из 1 мы k раз извлекли корень квадратный и всегда получали ± 1. Таким образом, мы не смогли найти свидетельств, что n составное число, и процесс останавливается.

Если $T \neq \pm 1$, мы не уверены, является ли n простым или составным числом, значит процесс будет продолжаться, и мы входим в цикл в строках 7–15. Рассмотрим шаг этого цикла. В строке 8 возводим T в квадрат. Если результат равен $+1$, мы определенно знаем, что тест Ферма пройден, потому что T остается 1 для последующих испытаний. Испытание квадратным корнем, однако, не пройдено. Поскольку T равно 1 на этом шаге и имело на предыдущем шаге другое значение, чем 1 (причина, почему мы не остановились на предыдущем шаге), n объявляют составным числом, и процесс останавливается. Если результат равен -1, мы знаем, что n в конечном счете пройдет тест Ферма. Мы знаем, что он пройдет испытание квадратным корнем, потому что T равно -1 в этом шаге и станет 1 на следующем шаге. Мы объявляем n псевдослучайным простым числом и останавливаем процесс. Если T имеет еще какое-либо значение, мы не уверены, имеем ли мы дело с простым числом, и процесс продолжается на следующем шаге. Остальные итерации цикла аналогичные.

В строке 16 мы возвращаем составное, потому что это означает что тест Ферма не пройден. Т. к. T содержит вычисленное значение $a^{n-1} \bmod n$, но в строках 10 и 13 выхода не последовало.

Вероятность того, что тест Миллера—Рабина, для случайно выбранного основания a, вернет «PossiblePrime» для составного числа n, не превосходит 0.25 [7].

Алгоритм 5.4 Тест Миллера—Рабина на простоту числа

```
1:  procedure MillerRabinTest(n, a)   ▷ Тест Миллера—Рабина для числа n по основанию a
2:      Найти m и k, такие что n − 1 = m · 2^k
3:      T ← a^m mod n
4:      if T = ±1 then
5:          return PossiblePrime
6:      end if
7:      for all i = 1, k − 1 do
8:          T ← T^2 mod n
9:          if T = +1 then
10:             return AuthenticallyNotPrime
11:         end if
12:         if T = −1 then
13:             return PossiblePrime
14:         end if
15:     end for
16:     return AuthenticallyNotPrime
17: end procedure
```

Пример 40. Применить тест Миллера—Рабина к числу 561.

Будем использовать основание 2, т. е. $a = 2$. Найдем m и k. Представим число 561 следующим образом $561 = 35 \cdot 2^4 + 1$. Отсюда $m = 35$, $k = 4$.

Вычислим $T = a^m \bmod n = 2^{35} \bmod 561 = 263 \neq \pm 1$. Вычисляем T^2 при $k = 1$, $T = 263^2 \bmod 561 = 166 \neq \pm 1$. Вычисляем T^2 при $k = 2$, $T = 166^2 \bmod 561 = 67 \neq \pm 1$. Вычисляем T^2 при $k = 3$, $T = 67^2 \bmod 561 = +1$. Возвращаем результат “составное” (тест Ферма пройден, тест квадратного корня не пройден).

Пример 41. Применить тест Миллера—Рабина к числу 27.

Будем использовать основание 2, т. е. $a = 2$. Найдем m и k. Представим число 27 следующим образом $27 = 13 \cdot 2^1 + 1$. Отсюда $m = 13$, $k = 1$.

Вычислим $T = a^m \bmod n = 2^{13} \bmod 27 = 11 \neq \pm 1$. В цикл не заходим, т.к. $k = 1$. Возвращаем результат “составное” (тест Ферма не пройден).

Пример 42. Применить тест Миллера—Рабина к числу 61.

Будем использовать основание 2, т. е. $a = 2$. Найдем m и k. Представим число 61 следующим образом $61 = 15 \cdot 2^2 + 1$. Отсюда $m = 15$, $k = 2$.

Вычислим $T = a^m \bmod n = 2^{15} \bmod 61 = 11 \neq \pm 1$. Вычисляем T^2 при $k = 1$, $T = 11^2 \bmod 61 = 60 = -1 \pmod{61}$. Возвращаем результат "простое" (все тесты пройдены).

Упражнение 46. Каково время работы алгоритма 5.4?

Упражнение 47. Продемонстрируйте работу алгоритма 5.4 для числа 89 по основанию 2.

Упражнение 48. Продемонстрируйте работу алгоритма 5.4 для числа 91 по основанию 2.

Упражнение 49. Как можно улучшить тест Миллера—Рабина?

5.2.4. AKS- алгоритм

В 2002 г. индийские ученые М. Агравал, Н. Каял и Н. Сахсена объявили, что они нашли алгоритм для испытания простоты чисел с полиномиальной сложностью времени разрядных операций.

Сравнение вида $a(x) \equiv b(x) \pmod{h(x), n}$ означает, что для многочленов $a(x)$ и $b(x)$ с коэффициентами из $\mathbb{Z}$, найдется многочлен $q(x)$ с коэффициентами из $\mathbb{Z}$ такой, что все коэффициенты многочлена $a(x) - b(x) - h(x)q(x)$ кратны n. В случае, когда $h(x) = 0$ будем иметь $a(x) \equiv b(x) \pmod{n}$.

Теорема 21. *Пусть n — нечетное натуральное число, r — простое число и выполнены условия: число n не делится ни на одно из чисел, меньших или равных r; порядок n в мультипликативной группе $\mathbb{Z}_p^*$ поля Z_p не меньше $\log_2^2(n)$; для всех a, $0 \leq a \leq r$, выполнено сравнение $(X + a)^n \equiv X^n + a \pmod{\frac{X^r-1}{X-1}, n}$. Тогда число n является простым.*

На основе теоремы AKS был построен соответствующий алгоритм 5.5. Отличительной особенностью алгоритма AKS является то, что это первый опубликованный алгоритм, который удовлетворяет всем четырем критериям.

1. **Универсальность**: Тест AKS может использоваться для проверки простоты любых чисел. Многие быстрые тесты предназначены для проверки чисел из ограниченного множества. Например, тест Люка—Лемера работает только для чисел Мерсенна, а тест Пепина применим лишь к числам Ферма.

2. **Полиномиальность**: Наибольшее время работы алгоритма ограничено полиномом от количества цифр в проверяемом числе.

Алгоритм 5.5 Тест Агравала—Каяла—Саксены на простоту числа

```
1: procedure AKSTest(n)  ▷ Тест Агравала — Каяла — Саксены для числа n > 1
2:     if n = a^b, a > 0, b > 1, a, b ∈ ℤ then
3:         return COMPOSITE
4:     end if
5:     Найти наименьшее r такое, что order_r(n) > log_2^2(n)
6:     if 1 < НОД(a, n) < n для некоторого a < n then
7:         return COMPOSITE
8:     end if
9:     if n < r then
10:        return PRIME
11:    end if
12:    for all a = 1, ⌊√(φ(r)) log_2(n)⌋ do
13:        T ← T^2 mod n
14:        if (X + a)^n ≢ X^n + a (mod X^r − 1, n) then
15:            return COMPOSITE
16:        end if
17:    end for
18:    return PRIME
19: end procedure
```

3. **Детерминизм**: Алгоритм гарантирует получение ответа. Вероятностные тесты, такие как тест Миллера—Рабина, могут проверить простоту числа за полиномиальное время, но при этом дают достоверного ответа, что число является простым.

4. **Безусловность**: Корректность теста AKS не зависит от каких-либо недоказанных гипотез.

Сегодня один из самых популярных тестов простоты чисел — комбинация теории делимости и тест Миллера—Рабина. При этом рекомендуются следующее шаги.

1. Выбрать нечетное целое число, потому что все четные целые числа (кроме 2) — явно составные числа.

2. Сделать некоторые тривиальные испытания теории делимости на некоторых известных простых числах, таких, как 3, 5, 7, 11, 13: так, чтобы убедиться, что не имеем дело с очевидным составным числом. Если выбранные простые числа не являются делителями, тогда перейти следующий шаг. Если выбранное число не прошло хотя бы один из этих тестов, вернуться на первый шаг и выбрать другое нечетное число.

3. Выбрать набор оснований для теста. Большое множество оснований предпочтительно.

4. Для каждого основания выполнить тест Миллера—Рабина. Если хотя бы один из тестов не прошел, то вернуться на первый шаг. Если все тесты пройдены, то объявите это число как сильное псевдопростое число.

Пример 43. Проверим на простоту число 4033 — составное число $4033 = 37 \cdot 109$. Подтверждает ли это рекомендованное испытание простоты чисел?

1. Выполним проверку согласно теории делимости . Проверим сначала числа $2, 3, 5, 7, 11, 17, 23$ — они не являются делителями числа 4033.

2. Выполним испытание Миллера—Рабина с основанием 2, тогда $4033 - 1 = 63 \cdot 2^6$, что означает $m = 63$ и $k = 6$. Начинаем итерации теста Миллера—Рабина. $T = 2^{63} \bmod 4033 = 3521$. Выполним первую итерацию цикла. $T = 3521^2 \bmod 4033 = 4032$. Результат теста "возможно простое".

3. Выполним тест с другим основанием, например $a = 3$. Начинаем итерации теста Миллера—Рабина. $T = 3^{63} \bmod 4033 = 3604$. Выполним первую итерацию цикла. $T = 3604^2 \bmod 4033 = 2556$. Выполним вторую итерацию цикла. $T = 2556^2 \bmod 4033 = 3709$. Выполним третью итерацию цикла. $T = 3709^2 \bmod 4033 = 118$. Выполним четвертую итерацию цикла. $T = 118^2 \bmod 4033 = 1825$. Выполним пятую итерацию цикла. $T = 1825^2 \bmod 4033 = 3400$. $k = 6$ цикл закончен. Результат 4033 — составное. Проверка завершена.

5.3. Разложение на множители

Разложение на множители — предмет непрерывного исследования в прошлом; и такие же исследования, вероятно, продолжатся в будущем. На сегодняшний день вопрос о существовании полиномиального алгоритма для разложения чисел на множители остается открытым. Убедимся, что всегда возможно разложить число на множители единственным образом с точностью до перестановки простых множителей.

Теорема 22 (Основная теорема арифметики)**.** *Каждое натуральное число $n > 1$ можно представить в виде $n = p_1 \cdot \cdots \cdot p_k$, где $p_1, \ldots, p_k$ — простые числа, причём такое представление единственно с точностью до порядка следования сомножителей.*

Доказательство. Существование. Докажем существование разложения числа n, учитывая, что оно уже доказано для любого другого числа, которое меньше n. Если n — простое, то существование доказано. Если n — составное, то оно может быть представлено в виде произведения двух чисел a и b, которые больше 1, но меньше n. Числа a и b либо являются

простыми, либо могут быть разложены в произведение простых (уже доказано ранее). Подставив их разложение в $n = a \cdot b$, получим разложение исходного числа n на простые. Существование доказано.

Единственность. Если разложение числа m на простые числа единственно, то каждый множитель m должен входить в это разложение. Пусть число m делится на p и при этом p — простое. Тогда можно представить исходное число как $m = p \cdot m_1$, где m_1 — натуральное число. Тогда разложение m — есть разложение числа m_1, с добавленным множителем p. По нашему предположению существует только одно разложение числа m, следовательно, p должно встретиться в нем.

Докажем единственность по индукции. Если n — простое, то единственность доказана. Если n — составное, то предположим, что существуют два разных представления числа n в произведение простых: $n = p_0 \cdot p_1 \cdot p_2 \cdot p_3 \cdot ... = q_0 \cdot q_1 \cdot q_2 \cdot q_3 \cdot ...$, где $p_i, q_i, i = 0, 1, 2, ...$ — простые числа.

В этих представлениях не могут встретиться два одинаковых простых числа, так как, если бы встретилось, то мы могли бы сократить на него и получили бы различные разложения числа, меньшего n, что противоречит предположению.

Пусть p_0— наименьший из простых множителей, которые встречаются в первом разложении. Так как n — составное, то существует еще хотя бы один множитель в разложении. И так как p_0 — был выбран как наименьший из всех множителей, то $n \geqslant p_0^2$. Во втором разложении аналогично: $n \geqslant q_0^2$. Так как $p_0 \neq q_0$, то одно из этих неравенств — строгое, следовательно, $p_0 q_0 < n$

Число $n - p_0 \cdot q_0$ — натуральное число, которое меньше n, следовательно, оно может быть представлено как произведение простых единственным способом. Так как n делится на p_0, то $n - p_0 \cdot q_0$ делится на p_0, следовательно, p_0 должно входить в разложение числа $n - p_0 \cdot q_0$. Аналогично q_0 — должно входить в разложение этого числа.

Таким образом получим, что $n = p_0 \cdot q_0 \cdot r_0 \cdot r_1 \cdot r_2 \cdot ...$, где $r_i, i = 0, 1, 2, ...$ — простые. Следовательно, число n делится на $p_0 q_0$, следовательно, $p_1 \cdot p_2 \cdot p_3 \cdot ...$ делится на q_0. Получим, что это невозможно, так как число $p_1 \cdot p_2 \cdot p_3 \cdot ... < n$ и q_0 не является одним из p_1, p_2, p_3, ... Следовательно, число n имеет единственное разложение на простые множители. □

5.3.1. Метод проверки делением

Самый простой и наименее эффективный алгоритм — метод разложения на множители с помощью деления. Перебираем все положительные целые числа, начиная с 2, для того чтобы найти одно, которое делит n. Если n составное, то делитель будет простым числом $p < \sqrt{n}$. Алгоритм 5.6 реализует этот метод.

Алгоритм 5.6 Метод проверки делением

```
1:  procedure FactorByDiv(n)
2:      a ← 2
3:      while a ≤ √n do
4:          if n mod a = 0 then
5:              return a
6:          end if
7:          a ← a + 1
8:      end while
9:      return n
10: end procedure
```

Метод проверки делением обычно хорош, если $n < 2^{10}$, но он неэффективен и неосуществим для разложения больших целых чисел. Сложность алгоритма $\mathcal{O}(\sqrt{n})$.

Упражнение 50. Используйте алгоритм проверки делением, чтобы найти сомножители числа 1233.

Упражнение 51. Используйте алгоритм проверки делением, чтобы найти сомножители 1523357784.

5.3.2. Метод Ферма

Метод Ферма разложения на множители (алгоритм 5.7) делит число n на два положительных целых числа (a и b — не обязательно простые числа) так, чтобы $n = a \cdot b$.

Алгоритм 5.7 Метод Ферма разложения на множители

```
1:  procedure FactorByFermat(n)
2:      x ← ⌈√n⌉
3:      while x ≤ n do
4:          w ← x² − n
5:          if (w - полный квадрат) then
6:              y ← √w
7:              a ← x + y
8:              b ← x − y
9:              return (a, b)
10:         end if
11:         x ← x + 1
12:     end while
13:     return FAIL
14: end procedure
```

Метод Ферма основан на факте, что если мы можем найти x и y такие, что $n = x^2 - y^2$, тогда мы имеем $n = x^2 - y^2 = (x + y)(x - y)$. Метод сводится к попытке найти два целых числа a и b, близкие друг к другу ($a \approx b$). Начинаем с наименьшего целого числа, большего, чем $x = \sqrt{n}$. Потом пробуем найти другое целое число y такое, чтобы выполнялось уравнение $y^2 = x^2 - n$. В каждой итерации мы должны рассмотреть, является ли $x^2 - n$ полным квадратом. Если мы находим такое значение для y, мы вычисляем a и b и выходим из цикла. Если мы не делаем этого, мы проводим следующую итерацию. Сложность метода Ферма является близкой к $\mathcal{O}(n)$.

5.3.3. $P - 1$ метод Полларда

Алгоритм 5.8 P-1 метод Полларда разложения на множители

procedure FACTORBYPOLLARDP1(n, a, B_1, B_2) ▷ Этап 1.
  $P \leftarrow \{p_1^{r_1}, p_2^{r_2}, ..., p_k^{r_k}, p_i \in \mathbb{P}, r_i \in \mathbb{N} | p_i^{r_i} < B_1\}$
  $M \leftarrow \prod_{p_i^{r_i} \in P_{B_1}} p_i^{r_i}$
  $d \leftarrow \gcd(n, a^M(B_1) - 1)$
  if $d \neq 1$ и $d \neq n$ **then**
    return d
  end if ▷ Этап 2.
  $b \leftarrow a^M(B_1) \bmod n$
  $Q \leftarrow \{q_1, q_2, ..., q_t, q_i \in \mathbb{P} | B_1 \leq q_i \leq B_2\}$
  $b \leftarrow a^M \bmod n$
  $c_0 \leftarrow b^{q_0} \bmod n$
  $d \leftarrow \gcd(n, c_0 - 1)$
  $i \leftarrow 1$
  while TRUE **do**
    if $d \neq 1$ и $d \neq n$ **then**
      return d
    end if
    if $i = t + 1$ **then**
      return FAIL
    end if
    $c_i \leftarrow b^{q_i} \bmod n$
    $d \leftarrow \gcd(n, c_0 - 1)$
    $i \leftarrow i + 1$
  end while
end procedure

В 1974 г. Д. Поллард разработал метод, который находит разложение числа n на простые числа.

Определение 24. Число называется *гладким* степени B, если все его простые делители в степенях, в которых они входят в разложение этого числа, меньше B .

Согласно малой теореме Ферма для любого простого числа p и для любого целого числа a такого, что a и p взаимно просты, справедливо: $a^{p-1} \equiv 1 \mod p$, более того $\forall M = (p-1)l$, $l \in \mathbb{N} \Rightarrow a^M \equiv 1 \mod p$. Последнее условие эквивалентно $a^{(p-1)} - 1 = pr$, для некоторого целого r . Отсюда, если p является делителем числа n , тогда p является делителем наибольшего общего делителя НОД$(n, a^M - 1)$.

Алгоритм состоит из двух стадий. Первая стадия. Пусть n гладкое степени B , и требуется найти делитель числа n. В первую очередь вычисляется число

$$M(B) = \prod_i p_i^{k_i},$$

где произведение ведётся по всем простым p_i в максимальных степенях $k_i : p_i^{k_i} < B$. Тогда искомый делитель $q = \text{НОД}(a^{M(B)} - 1, n)$.

Возможно два случая, в которых приведенный выше алгоритм не даст результата. В случае, когда НОД$(a^{M(B)} - 1, n) = n$ точно можно сказать, что у n есть делитель, являющийся гладким степени B и проблему должен решить иной выбор a. В более частом случае, когда НОД$(a^{M(B)} - 1, n) = 1$ стоит перейти ко второй стадии алгоритма, которая значительно повышает вероятность результата, хотя и не гарантирует его.

Для выполнения второй стадии необходимо зафиксировать границы $B_1 = B$, $B_2 \gg B$, обычно $B_2 \leq B^2$. Вторая стадия алгоритма находит делители n, такие что $p - 1 = q \cdot f$, где f — гладкое степени B, а q простое, такое что $B_1 < q < B_2$.

Для дальнейшего нам потребуется вектор из простых чисел q_i от B_1 до B_2, из которого легко получить вектор разностей между этими простыми числами $D = (D_1, D_2, ...)$, $D_i = q_{i+1} - q_i$, причем D_i — относительно небольшие числа, и $D_i \in \Delta$, где Δ — конечное множество. Для ускорения работы алгоритма полезно предварительно вычислить все $b^{\delta_i}, \forall \delta_i \in \Delta$ и пользоваться уже готовыми значениями. Теперь необходимо последовательно вычислять $c_0 = b \bmod n$, $c_i = c_{i-1}^{\delta_i} \bmod n$, где $b = a^M(B_1) \bmod n$, вычисленное в первой стадии, на каждом шаге считая $Q = \text{НОД}(c_i - 1, n)$. Как только $Q \neq 1$, можно прекращать вычисления.

5.3.4. ρ-метод Полларда

Пробное деление на каждое целое число вплоть до B гарантирует, что будет полностью разложено любое число вплоть до B^2. Представленная

ниже процедура позволяет разложить любое число вплоть до B^4, выполнив тот же объем работы. Поскольку эта процедура носит лишь эвристический характер, ничего нельзя утверждать наверняка: ни о времени ее работы, ни о том, что она действительно достигнет успеха. Несмотря на это данная процедура оказывается очень эффективной на практике. Другое преимущество процедуры состоит в том, что в ней используется лишь фиксированное количество памяти.

Основная идея поиска множителя q составного числа n заключается в следующем. Мы организуем последовательность $x_i = f(x_{i-1})$. А далее мы хотим найти два таких элемента x_j, x_k в этой последовательности, чтобы $x_j \equiv x_k \pmod{q}$. Ниже мы обсуждаем, почему это скорее всего произойдет. А здесь отметим, что если нам удалось найти такие x_j, x_k, то делитель числа n может быть найден как НОД$(x_j - x_k, n)$.

Алгоритм 5.9 ρ-метод Полларда разложения на множители

```
1:  procedure FactorByPollardRho(n)
2:      i ← 1
3:      x_1 ← Random(0, n − 1)
4:      y ← x_1
5:      k ← 2
6:      while TRUE do
7:          i ← i + 1
8:          x_i ← (x_{i−1}^2 − 1) mod n
9:          d ← gcd(y − x_i, n)
10:         if d ≠ 1 и d ≠ n then
11:             return d
12:         end if
13:         if i = k then
14:             y ← x_i
15:             k ← 2k
16:         end if
17:     end while
18: end procedure
```

Опишем работу этой процедуры. В строках 1–2 переменной i присваивается начальное значение 1, а переменной x_i — случайное значение из $\mathbb{Z}_n$. Итерации цикла **while,** который начинается в строке 6, продолжаются до бесконечности в поисках множителей числа n. Выход из этого цикла возможен только в строке 11, и в этом случае нетривиальный множитель числа n будет найден. Возникает вопрос: почему этот цикл завершится. Рассмотрим последовательность x_i, генерируемую в строке 8 по формуле $x_i \leftarrow (x_{i-1}^2 - 1) \bmod n$. Для выхода из цикла необходимо,

чтобы множитель разности $y - x_i$ совпал со множителем n. При этом значение y определяется в строках 13–16 и является по сути прошлым значением x_i. Пусть q наименьший множитель числа n, тогда рассмотрим последовательность $u_i = x_i \bmod q$. Обратим внимание, что в строке 8 формула для x_i построена таким образом, что выполняется соотношение $u_{i+1} = x_{i+1} \bmod q = (x_i^2 - 1) \bmod n \bmod q = (x_i^2 - 1) \bmod q = \Big((x_i \bmod q)^2 - 1\Big) \bmod q = \big(u_i^2 - 1\big) \bmod q$. Таким образом, последовательность u_i построена по такой же схеме, что и последовательность x_i. Элементы последовательности x_i могут принимать лишь фиксированное количество значений, а значит как только выполнится $x_i = x_j$, то последовательность x_i зацикливается. Аналогично последовательность u_i зациклится раньше. Что произойдет, когда последовательность u_i зациклится? В этом случае $u_i = u_j$, а значит $(x_i - x_j) \bmod q = 0$, а следовательно, $x_i - x_j = kq$, $k \in \mathbb{Z}$. В строке 9 с помощью НОД мы можем найти общий делитель в разнице $x_i - x_j$ и n. Постоянное удлинение периода поиска в строке 15 гарантирует обнаружение цикла.

Несмотря на то, что эти последовательности x_i генерируются по детерминированной формуле, мы можем рассматривать их как случайные. Всего в последовательности u_i только p элементов, поэтому согласно парадоксу дней рождения нам потребуется перебрать в среднем $\sqrt{q}$ элементов до встречи двух одинаковых. Учитывая, что $q^2 \leq n$, то время работы алгоритма $\mathcal{O}(n^{\frac{1}{4}})$.

6. ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ

Рассмотрим одну из широко используемых в криптографии группу — группу точек эллиптической кривой.

Определение 25. *Эллиптическая кривая* над полем K — это множество точек поля (x, y), удовлетворяющих уравнению $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ вместе с точкой на бесконечности.

Определение 26. *Характеристикой поля* K называется наименьшее ненулевое натуральное число n такое, что $n \cdot 1 = 0$, если же такого n не существует, то характеристика равна нулю.

Если характеристика равна 2, то эллиптическая кривая с помощью замены координат может быть приведена к виду $y^2 + y = x^3 + ax + b$ или $y^2 + xy = x^3 + ax^2 + b$. Если характеристика равна 3, то эллиптическая кривая приводится к виду $y^2 = x^3 + a_2x^2 + a_4x + a_6$. Если характеристика поля K не равна 2 или 3, то уравнение с помощью замены координат приводится к канонической форме. В этом виде в дальнейшем мы и будем рассматривать уравнение эллиптической кривой

$$y^2 = x^3 + ax + b. \tag{6.1}$$

Примеры эллиптических кривых на действительной плоскости представлены на рис. 6.1.

Поскольку в дальнейшем нам потребуется возможность находить касательную к точкам эллиптической кривой, то будем рассматривать только кривые без особых точек. Алгебраически это значит, что дискриминант $\Delta = -16(4a^3 + 27b^2)$ не должен быть равен нулю. Если кривая не имеет особых точек, то её график имеет две части, если дискриминант положителен, и одну — если отрицателен. Например, для графиков на рис. 6.1 в случае а) дискриминант равен 1728, а в случае д) он равен -368.

В дальнейшем будем обозначать эллиптическую кривую без особых точек $E(a, b)$.

Упражнение 52. Рассчитайте дискриминанты для всех кривых, приведенных на рис. 6.1. Есть ли среди них кривые с особыми точками?

6.1. Группа точек эллиптической кривой

Пусть $E(a, b)$ эллиптическая кривая, заданная уравнением $y^2 = x^3 + ax + b$, над полем вещественных чисел. Выберем две произвольные точки

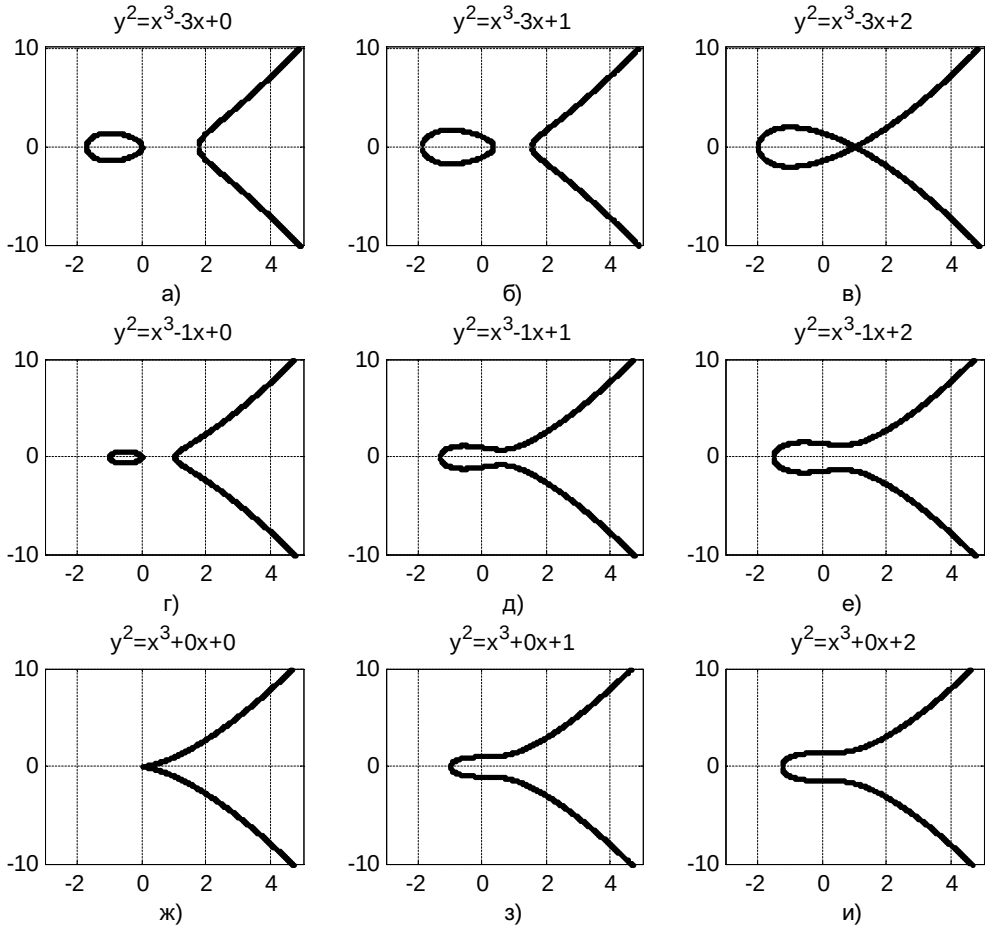


Рис. 6.1. Примеры эллиптических кривых

на этой кривой $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(a, b)$. Рассмотрим множество точек кривой $E(a, b)$ и точку O — точку, удаленную на бесконечности. Потребуем, чтобы сумма точек эллиптической кривой лежащих на одной прямой являлась нулем

$$P + Q + R = O. \tag{6.2}$$

Определим точку O как тождественный элемент по сложению («нулевой элемент») группы точек. Обратим внимание, что точки P и $-P$ имеют одинаковые x-координаты, а их y-координаты различаются только знаком, т.е. $-(x, y) = (x. - y)$. Из (6.1) сразу следует, что $(x, -y)$, – также точка на $E(a, b)$. Из (6.2) следует, что $P + Q = -R$. Найдем координаты точки $-R$.

Пусть дана кривая $y^2 = x^3 + ax + b$ над полем K (чья характеристика не равна ни 2, ни 3) и точки $P = (x_P, y_P)$ и $Q = (x_Q, y_Q)$ на кривой, допустим, что $x_P \neq x_Q$. Тогда координаты точки $-R = P + Q$ выражаются

следующим образом

$$\begin{cases} \lambda = \frac{y_P - y_Q}{x_P - x_Q}, \\ x_{-R} = \lambda^2 - x_P - x_Q, \\ y_{-R} = -y_P + \lambda(x_P - x_R). \end{cases} \tag{6.3}$$

Если $x_P = x_Q$, то возможны три варианта: если $y_P = -y_Q$, то сумма определена как O. Если $x_P = x_Q$ и $y_P = y_Q \neq 0$, то $R = P + P = 2P = (x_R, y_R)$ определяется как

$$\begin{cases} \lambda = \frac{3x_P^2 + a}{2y_P}, \\ x_{-R} = \lambda^2 - 2x_P, \\ y_{-R} = -y_P + \lambda(x_P - x_R). \end{cases} \tag{6.4}$$

Если $x_P = x_Q$ и $y_P = y_Q = 0$, то $P + P = 0$.

Таким образом для множества точек эллиптической кривой выполняются все аксиомы группы: замкнутость, ассоциативность, коммутативность, существование нейтрального элемента, существование обратного элемента.

Определение 27. *Порядком точки P на эллиптической кривой $E(a, b)$* называется такое минимальной число $n \in \mathbb{N}$, что

$$nP = \underbrace{P + P + \ldots + P}_{n} = O.$$

В случае, если такого n не существует, то говорят о точке бесконечного порядка.

Пример 44. Найти порядок точки $P = (2, 3)$ на эллиптической кривой $E(0, 1)$.

Кривая $E(0, 1)$ описывается уравнением $y^2 = x^3 + 1$. Проверим, что точка P принадлежит этой кривой, для этого надо проверить справедливость равенства $3^2 = 2^3 + 1$. Равенство выполняется, следовательно, $P \in E(0, 1)$. Применяя формулы (6.4) найдем точку $2P = P + P = (0, 1)$. Снова применим формулу (6.4) и найдем точку $4P = 2(2P) = 2P + 2P = (0, -1)$. Заметим, что $4P = -2P$, а следовательно, $6P = 0$. Поэтому порядок точки P может быть равен 2, 3 или 6. Но $2P = (0, 1) \neq O$, а если бы P имела порядок 3, то $4P = P$, что неверно. Следовательно, P имеет порядок 6.

Упражнение 53. Выведите выражения для координат точки $P + Q$ через координаты точек P и Q для случая $P \neq Q$.

Упражнение 54. Выведите выражения для координат точки $P + Q$ через координаты точек P и Q для случая $P = Q$.

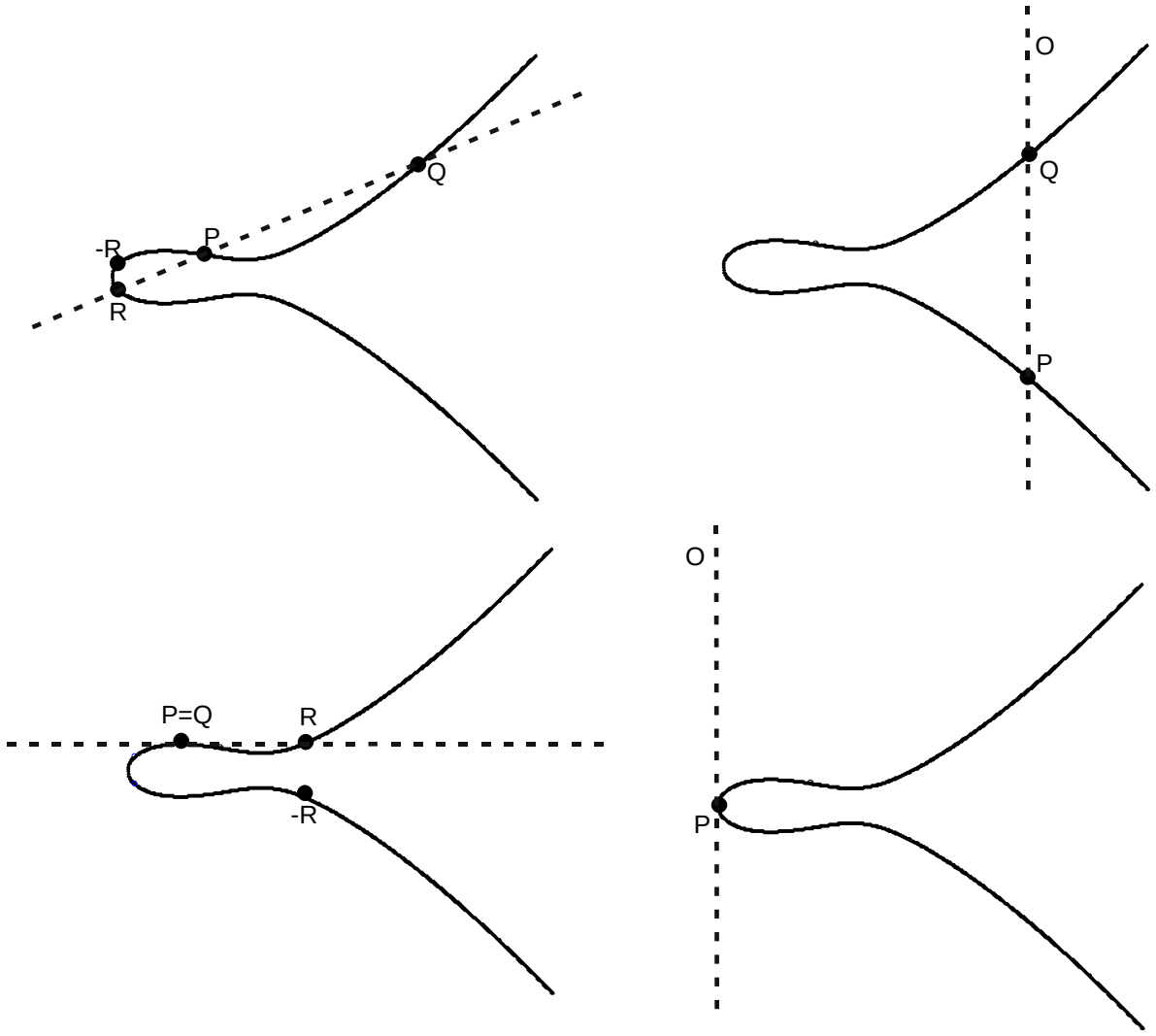


Рис. 6.2. Сложение точек эллиптической кривой

Упражнение 55. Докажите, что группа точек эллиптической кривой абелева.

Упражнение 56. Существует ли эллиптическая кривая, все точки которой имеют порядок 3.

6.2. Эллиптические кривые над конечными полями

Ранее были рассмотрены эллиптические кривые, определенные над бесконечным полем вещественных чисел. Попробуем перенести эти определения в конечное поле $\mathbb{Z}_p$. Также будут существовать точки, описываемые парой элементов поля, а формулы выполнения сложения не поменяются, за исключением необходимости проводить все вычисления по правилам поля $\mathbb{Z}_p$. Эллиптическую кривую над полем $\mathbb{Z}_p$ будем обозначать как $E_p(a, b)$. Ее уравнение аналогично уравнению кривой над полем

действительных чисел

$$y^2 \equiv x^3 + ax + b \pmod{p}. \tag{6.5}$$

Параметрами кривой $E_p(a, b)$ являются дискриминант $\Delta = 4a^3+27b^2$, который должен быть не сравним с p и инвариант

$$J(E) \equiv 1728\frac{4a^3}{4a^3 + 27b^2} \pmod{p}.$$

Если $J(E) \notin \{0, 1728\}$, то параметры кривой a, b определяются по формулам

$$a \equiv 3k \pmod{p},\ b \equiv 2k \pmod{p},$$

где $k \equiv \frac{J(E)}{1728-J(E)} \pmod{p}$.

Таблица 6.1
Группа точек на эллиптической кривой $E_{11}(-2, 5)$

Номер	X	Y
1	0	4
2	0	7
3	1	2
4	1	9
5	2	3
6	2	8
7	3	2
8	3	9
9	6	0
10	7	2
11	7	9
12	9	1
13	9	10
14	∞	∞

Пример 45. Рассмотрим, что представляет собой эллиптическая кривая $y^2 = x^3 - 2x + 5$ над полем $\mathbb{Z}_{11}^*$. Элементы этого поля являются числа от 0 до 10, все операции выполняются по модулю 11. В качестве x мы можем рассматривать все значения от 0 до 10, после чего для каждого значения необходимо вычислить $x^3 - 2x + 5 \bmod 11$ и, если это число квадратичный вычет, вычислить квадратичный корень по модулю 11, пары $(x, \sqrt{x^3 - 2x + 5} \bmod n)$ и $(x, n - \sqrt{x^3 - 2x + 5} \bmod n)$ будут являться точками на эллиптической кривой $E_{11}(-2, 5)$. Список всех точек кривой приведен в таблице 6.1. Таким образом группа включает в себя 14 точек, что соответствует теореме Хассе.

Пример 46. Определим порядок точки $G = (7, 2)$ на кривой $E_{11}(-2, 5)$. Для этого вычислим nG. Результаты вычислений представлены в таблице 6.2. Порядок точки G равен 7.

Таблица 6.2
Степени точки $(7, 2)^n$ на кривой $E_{11}(-2, 5)$

n	1	2	3	4	5	6	7
X	7	0	9	9	0	7	∞
Y	2	7	1	10	4	9	∞

Для оценки порядка группы точек эллиптической кривой удобно использовать теорему Хассе [14].

Теорема 23 (Хассе)**.** Пусть N — число точек эллиптической кривой, определенной над конечным полем F_q, тогда

$$|N - q - 1| \leq 2\sqrt{q}.$$

Упражнение 57. Определите порядок точки $(7, 0)$ на $E_{13}(1, 1)$.

Упражнение 58. Определите порядок точки $(11, 2)$ на $E_{13}(1, 1)$.

7. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

В 70-х годах прошлого века удалось построить алгоритмы ассиметричного шифрования (рис. 7.1), которые решали основную проблему симметричных криптосистем — проблему передачи ключа. В этих схемах используется два различных ключа: один известный всем или открытый ключ, необходимый для шифрования и один закрытый, известный только Получателю, необходимый для дешифрования.

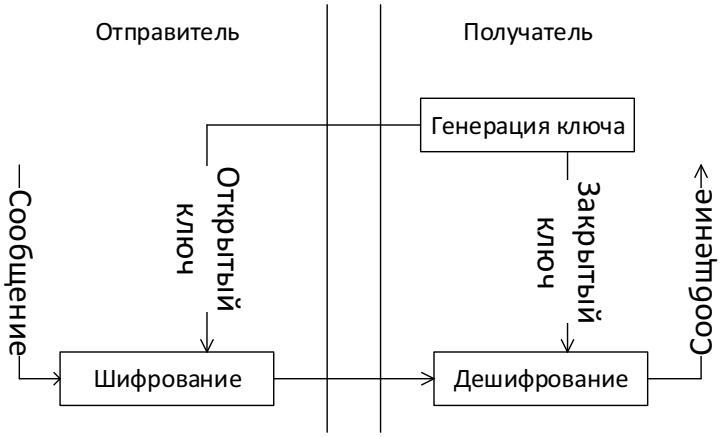


Рис. 7.1. Схема ассиметричного шифрования

7.1. Алгоритм обмена ключами Диффи-Хеллмана

Алгоритм Диффи-Хеллмана — криптографический протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя незащищенный от прослушивания канал связи (рис. 7.2), был предложен У. Диффи и М. Хеллманом в 1976 г. Полученный ключ может использоваться для шифрования дальнейшего обмена с помощью алгоритмов симметричного шифрования.

Предположим, существует два абонента: Отправитель и Получатель. Обоим абонентам известны некоторые два числа g и p, которые не являются секретными и могут быть известны также другим заинтересованным лицам. Число p должно быть большим простым числом, а число g первообразным корнем по модулю p.

Этап генерации ключей. Для того, чтобы создать неизвестный более никому секретный ключ, оба абонента генерируют большие случайные числа: Отправитель — число a, Получатель — число b. Затем Отправитель вычисляет значение

$$A = g^a \bmod p$$

и пересылает его Получатель, а Получатель вычисляет

$$B = g^b \bmod p$$

и передаёт Отправитель. Предполагается, что злоумышленник может получить оба этих значения, но не модифицировать их (то есть у него нет возможности вмешаться в процесс передачи).

Далее Отправитель на основе имеющегося у него a и полученного по сети B вычисляет значение

$$B^a \bmod p = g^{ab} \bmod p.$$

Получатель на основе имеющегося у него b и полученного по сети A вычисляет значение

$$A^b \bmod p = g^{ab} \bmod p.$$

У Отправителя и Получателя получилось одно и то же число

$$K = g^{ab} \bmod p.$$

Его они и могут использовать в качестве секретного ключа, поскольку здесь злоумышленник встретится с практически неразрешимой (за разумное время) проблемой вычисления K по перехваченным $g^a \bmod p$ и $g^b \bmod p$, если числа p, a, b выбраны достаточно большими.

Отправитель		Получатель
Подготовка, выбор параметров		
Выбираем случайное больше простое p и g — первообразный корень по модулю p	$\overset{p,g}{\Longrightarrow}$	
Генерация и обмен ключами		
Выбираем большое случайное число a		Выбираем большое случайное число b
Вычисляем $A = g^a \bmod p$	$\overset{A}{\Longrightarrow}$	
	$\overset{B}{\Longleftarrow}$	Вычисляем $B = g^b \bmod p$
Вычисляем общий секретный ключ $K = B^a \bmod p = g^{ab} \bmod p$		Вычисляем общий секретный ключ $K = A^b \bmod p = g^{ab} \bmod p$

Рис. 7.2. АлгоритмДиффи-Хеллмана

Пример 47. Пусть $p = 7$, а $g = 3$. Отправитель выбрал $a = 2$, а получатель $b = 5$. Отправитель вычисляет значение $A = 3^2 \bmod 7{=}2$. Получатель вычисляет значение $B = 3^5 \bmod 7 = 5$. Общий ключ $K = 2^5 \bmod 7 = 5^2 \bmod 7 = 4$. Для взлома злоумышленнику потребуется решить задачу дискретного логарифмирования $3^a \equiv 2 \pmod 7$.

Упражнение 59. Предложите модификацию алгоритма Диффи-Хеллмана для трех участников.

7.2. Алгоритм обмена ключами Диффи-Хеллмана на основе эллиптической кривой

Алгоритм обмена ключами Диффи-Хеллмана с использованием эллиптических кривых (рис. 7.3) может быть выполнен следующим образом. Сначала выбирается простое число p и параметры a и b для эллиптической кривой в уравнении (6.5). Эти параметры задают группу точек на кривой $E_p(a,b)$. Далее выбирается генерирующая точка $G = (x_G, y_G) \in E_p(a,b)$. При выборе G важно, чтобы её порядок оказался очень большим простым числом. Параметры $E_p(a,b)$ и G криптосистемы являются параметрами, известными всем участникам. Обмен ключами между Отправителем и Получателем можно провести по следующей схеме:

1. Отправитель выбирает целое число n_A, меньшее порядка точки G. Это число будет личным ключом Отправителя. Затем Отправитель генерирует открытый ключ $P_A = n_A G$. Открытый ключ представляет собой некоторую точку из $E_p(a,b)$.

2. Точно так же Получатель выбирает личный ключ n_B и вычисляет свой открытый ключ $P_B = n_B G$.

3. Отправитель генерирует общий секретный ключ $K = n_A P_B$, а Получатель генерирует секретный ключ $K = n_B P_A$.

Две формулы в п. 3 дают один и тот же результат, поскольку $n_A P_B = n_A n_B G = n_B n_A G = n_B P_A$. Чтобы взломать эту схему, противник должен будет вычислить n_A по данным G и $n_A G$, что предполагается трудной задачей.

Пример 48. Рассмотрим эллиптическую кривую $E_{11}(-2,5)$, т. е. кривая описывается соотношением

$$y^2 \equiv x^3 - 2x + 5 \pmod{11}.$$

Пусть выбрана точка $G = (7,2)$, которая лежит на кривой $E_{11}(-2,5)$, т.к. $2 \cdot 2 \equiv 49 \cdot 7 - 14 + 5 \pmod{11}$. Можно подсчитать, что порядок точки G равен 7. Личный ключ участника A равен $n_A = 3$ и ему соответствует точка $P_A = 3G = (9,1)$. Личным ключом пользователя B является число $n_B = 5$ и ему соответствует точка $P_B = 5G = (10,4)$. Общий ключ будет иметь вид $n_A P_B = n_A n_B G = n_B n_A G = n_B P_A = 3 \cdot 5 \cdot G = (2,1)$.

Упражнение 60. Предложите криптоатаку при условии, что порядок G — маленькое простое число.

7.3. Алгоритм рюкзака для шифрования

Использование проблемы рюкзака для шифрования с открытым ключом было предложено Р. Мерклом и М. Хеллманом в 1978 г. Безопасность

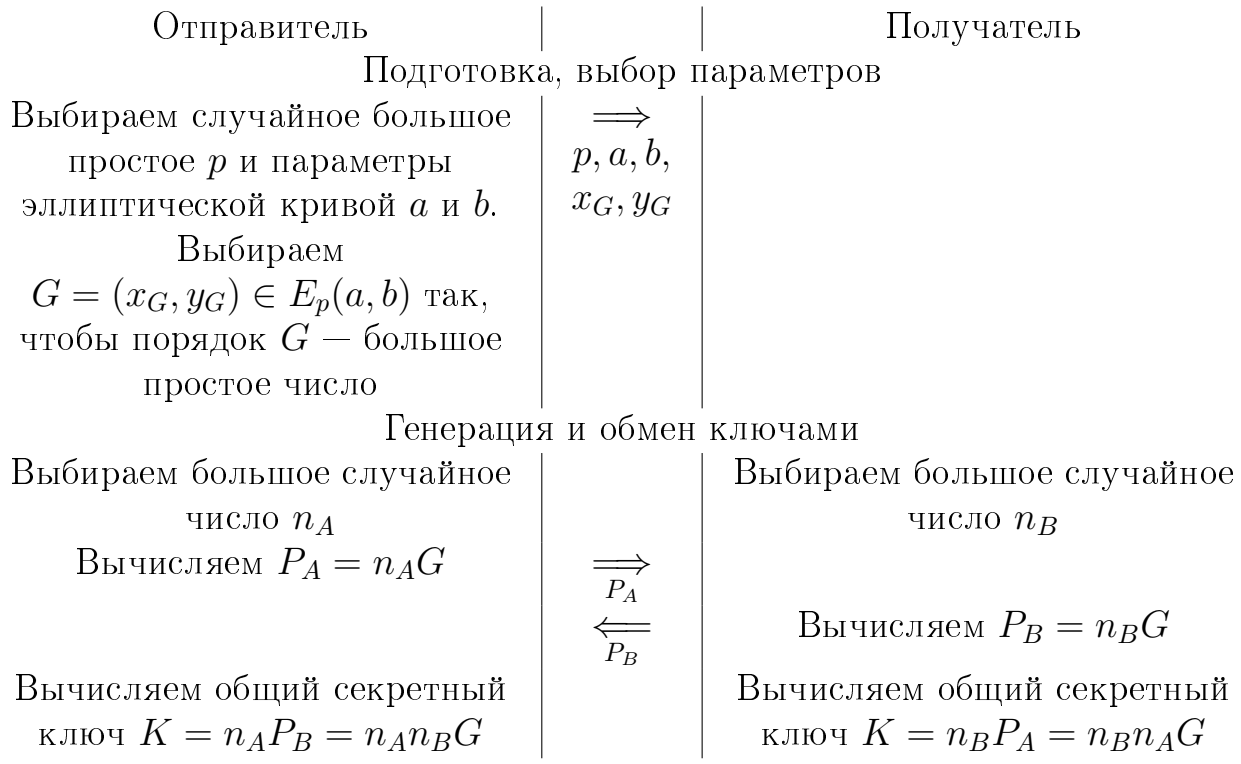


Рис. 7.3. Алгоритм Диффи-Хеллмана на эллиптических кривых

алгоритмов рюкзака опирается на проблему рюкзака, NP-полную проблему. Хотя позже было обнаружено, что этот алгоритм небезопасен, его стоит изучить, так как он демонстрирует возможность применения NP-полной проблемы в криптографии с открытыми ключами.

Проблема рюкзака заключается в следующем. Дано n предметов различной массы $M_1, M_2, ..., M_n$, можно ли положить некоторые из этих предметов в рюкзак так, чтобы масса рюкзака стала равна определенному значению S? Формально требуется отыскать бинарный вектор b такой что

$$S = b_1 M_1 + b_2 M_2 + ... + b_n M_n,$$

где b_i, $i = \overline{1, n}$ может быть либо нулем, либо единицей. Единица показывает, что предмет кладут в рюкзак, а ноль — что не кладут.

Например, массы предметов могут иметь значения 1, 5, 6, 11, 14 и 20. Можно упаковать рюкзак так, чтобы его масса стала равна 22, использовав массы 5, 6 и 11. Невозможно упаковать рюкзак так, чтобы его масса была равна 24. В общем случае время, необходимое для решения этой проблемы, с ростом количества предметов растет экспоненциально.

Существует интересная особенность в задаче о рюкзаке. Если перечень масс представляет собой сверхвозрастающую последовательность, то полученную проблему рюкзака легко решить.

Определение 28. *Сверхвозрастающая последовательность* — это последовательность, в которой каждой член больше суммы всех предыдущих членов.

Например, последовательность $1, 3, 6, 13, 27, 52$ является сверхвозрастающей, а $1, 3, 4, 9, 15, 25$ — нет. Решение сверхвозрастающего рюкзака найти легко. Возьмем полный вес и сравним его с самым большим числом последовательности . Если полный вес меньше, чем это число, то его не кладут в рюкзак . Если полный вес больше или равен этому числу, то оно кладется в рюкзак . Уменьшим массу рюкзака на это значение и перейдем к следующему по величине числу последовательности . Будем повторять, пока процесс не закончится.

Например, пусть полный вес рюкзака — 70, а последовательность весов $2, 3, 6, 13, 27, 52$. Самый большой вес 52, меньше 70, поэтому кладем 52 в рюкзак. Вычитая 52 из 70, получаем 18. Следующий вес 27, больше 18, поэтому 27 в рюкзак не кладется. вес 13 меньше 18, поэтому кладем 13 в рюкзак. Вычитая 13 из 18, получаем 5. Следующий вес 6, больше 5, поэтому 6 не кладется в рюкзак. Продолжение этого процесса покажет, что и 2, и 3 кладутся в рюкзак, и полный вес уменьшается до 0, что сообщает о найденном решении.

Процесс шифрования показан в таблице 7.1. Предметы из множества выбираются с помощью блока открытого текста, по длине равного количеству предметов (биты открытого текста соответствуют значениям b), а шифротекст является полученной суммой. Для дешифрования в случае обычного рюкзака требуется решать сложную проблему о рюкзаке, в случае сверхвозрастающей последовательности весов — эта задача превращается в легкую задачу. Закрытый ключ помогает свести общую задачу к задаче со сверхвозрастающими весами. Перейдем к описанию алгоритма.

Таблица 7.1

Пример шифрования в алгоритме рюкзака

Открытый текст	1 1 1 0	0 1 0 1	0 0 0 0	0 1 1 0
Рюкзак	1 5 6 11	1 5 6 11	1 5 6 11	1 5 6 11
Шифротекст	1+5+6=12	5+11=16	0=0	5+6=11

Этап — генерация ключа. Чтобы зашифровать n-битное сообщение, выберем сверхвозрастающую последовательность из n ненулевых натуральных чисел $w = (w_1, w_2, ..., w_n)$. Далее случайным образом выберем целые взаимно простые числа q и r такие, что

$$q > \sum_{i=1}^{n} w_i.$$

Число q выбирается таким образом, чтобы гарантировать единственность шифротекста. Теперь построим последовательность $\beta = (\beta_1, \beta_2, ..., \beta_n)$, где каждый элемент вычисляется по формуле

$$\beta_i = rw_i \bmod q.$$

β будет открытым ключом, в то время как закрытый ключ образует тройка (w, q, r).

Этап — шифрование. Чтобы зашифровать n-битное сообщение $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$, где$\alpha_i \in 0, 1$, вычислим число c, которое и будет шифротекстом по формуле

$$c = \sum_{i=1}^{n} \alpha_i \beta_i.$$

Этап — дешифрование. Чтобы прочитать исходный текст, получатель должен определить биты сообщения α_i, которые удовлетворяли бы формуле

$$c = \sum_{i=1}^{n} \alpha_i \beta_i.$$

Такая задача была бы NP-сложна в случае, если бы β_i были случайно выбранными значениями. Однако значения β_i были выбраны таким образом, что расшифровка сводится к простой задаче при условии, что закрытый ключ (w, q, r) известен.

Ключ к определению исходного текста заключается в том, чтобы найти целое число s, которое является мультипликативным обратным r по модулю q. Далее получатель шифротекста вычисляет

$$c' \equiv cs \pmod{q}.$$

Отсюда

$$c' \equiv cs \equiv \sum_{i=1}^{n} \alpha_i \beta_i s \pmod{q}.$$

Из того, что $rs \bmod q = 1$ и $\beta_i = rw_i \bmod q$, следует $\beta_i s \equiv w_i rs \equiv w_i \pmod{q}$. Тогда

$$c' \equiv \sum_{i=1}^{n} \alpha_i w_i \pmod{q}.$$

Сумма всех w_i меньше, чем q. Отсюда значение

$$\sum_{i=1}^{n} \alpha_i w_i$$

также находится в интервале $[0, q-1]$. Таким образом, получатель должен определить α_i из условия, что

$$c' = \sum_{i=1}^{n} \alpha_i w_i.$$

А это уже простая задача, поскольку w — сверхвозрастающая последовательность. Пусть w_k – наибольший элемент в w. Если $w_k > c'$, то $\alpha_k = 0$; если $w_k \leq c'$, то $\alpha_k = 1$. Далее вычитаем произведение $w_k\alpha_k$ из c' и повторяем эти шаги до тех пор, пока не вычислим α.

Пример 49. Пусть $w = 2, 7, 11, 21$ — сверхвозрастающая последовательность. Она является основой для генерации закрытого ключа. Далее выберем простое число q, превосходящее значение суммы $\sum w = 41$. Пусть $q = 47$. Выберем также число r из интервала $[1, q)$ и пусть $r = 30$. Тройка w, q и r образует закрытый ключ. Чтобы сгенерировать открытый ключ, построим последовательность β, умножая каждый элемент из последовательности w на r по модулю q.

$$2 \cdot 30 \bmod 47 = 13; \quad 7 \cdot 30 \bmod 47 = 22;$$
$$11 \cdot 30 \bmod 47 = 1; \quad 21 \cdot 30 \bmod 47 = 19.$$

Получим $\beta = (13, 22, 1, 19)$. Последовательность β образует открытый ключ. Пусть Отправитель хочет зашифровать двоичный код 0110. Он умножает каждый бит на соответствующее число из последовательности β, а сумму значений $a = 0 \cdot 13 + 1 \cdot 22 + 1 \cdot 1 + 0 \cdot 19 = 23$ отправляет получателю. Чтобы расшифровать сообщение, Получатель умножает полученное им значение на мультипликативное обратное r по модулю q: $23 \cdot 11 \bmod 47 = 18$.

После этого Получатель раскладывает 18 следующим образом. Сначала он выбирает наибольший элемент из w, который меньше, чем 18, и вычисляет их разность. Далее он выбирает следующий наибольший элемент, который меньше, чем полученная разность, и повторяет эти действия, пока разность не станет равной нулю. Элементы, которые были выбраны из w , будут соответствовать 1 в двоичной записи исходного текста.

Упражнение 61. Докажите, что задача рюкзака со сверхвозрастающей последовательностью решается за линейное время.

7.4. Алгоритм шифрования RSA

Наиболее известным, наверное, является алгоритм с открытым ключом, который можно использовать для шифрования и цифровых подписей — RSA. Названный в честь трех изобретателей: Р. Ривеста, А.

Шамира и Л. Эдлмана. Этот алгоритм (рис. 7.4) многие годы противостоит интенсивному криптоанализу. Хотя криптоанализ ни доказал, ни опроверг безопасность RSA, он, по сути, обосновывает уровень доверия к алгоритму. Безопасность RSA основана на трудности разложения на множители больших чисел. Открытый и закрытый ключи являются функциями двух больших (100–200 разрядов или даже больше) простых чисел. Предполагается, что восстановление открытого текста по шифротексту и открытому ключу эквивалентно разложению на множители двух больших чисел.

Для генерации двух ключей используются два больших случайных простых числа p и q. Для максимальной безопасности выбирают p и q равной длины. Рассчитывается произведение $n = p \cdot q$. Затем случайным образом выбирается ключ шифрования e, такой что e и $\varphi(n) = (p-1)(q-1)$ являются взаимно простыми числами. Расширенный алгоритм Евклида используется для вычисления ключа дешифрирования d, такого что $d = e^{-1} \pmod{\varphi(n)}$

Заметим, что d и n также взаимно простые числа. Числа e и n — это открытый ключ, а число d — закрытый. Два простых числа p и q больше не нужны. Они должны быть отброшены, но не должны быть раскрыты.

Для шифрования сообщения m оно сначала разбивается на цифровые блоки, меньшие n (для двоичных данных выбирается самая большая степень числа 2, меньшая n). Формула шифрования выглядит так

$$c_i = m_i^e \bmod n.$$

Для расшифровки сообщения возьмем каждый зашифрованный блок c_i и вычислим $m_i = c_i^d \bmod n$. Так как

$$e \cdot d = 1 + k \cdot \varphi(n),$$

$$c_i^d = (m_i^e)^d = m_i^{ed} = m_i^{1+k\cdot\varphi(n)} = m_i \cdot \left(m_i^{\varphi(n)}\right)^k = m_i.$$

Пример 50. Зашифруем и расшифруем сообщение «CAB» по алгоритму RSA. Для простоты возьмем небольшие числа — это сократит наши расчеты.

Выберем$p = 3$ and $q = 11$. Определим $n = 3 \cdot 11 = 33$. Найдем $(p-1) \cdot (q-1) = 20$. Следовательно, e будет равно, например, $e = 3$.

Выберем число d по следующей формуле: $(d \cdot 3) \bmod 20 = 1$. Значит $d = 7$.

Представим шифруемое сообщение как последовательность чисел в диапазоне от 0 до 32. Буква $A = 1$, $B = 2, C = 3$. Теперь зашифруем сообщение, используя открытый ключ $7, 33$.

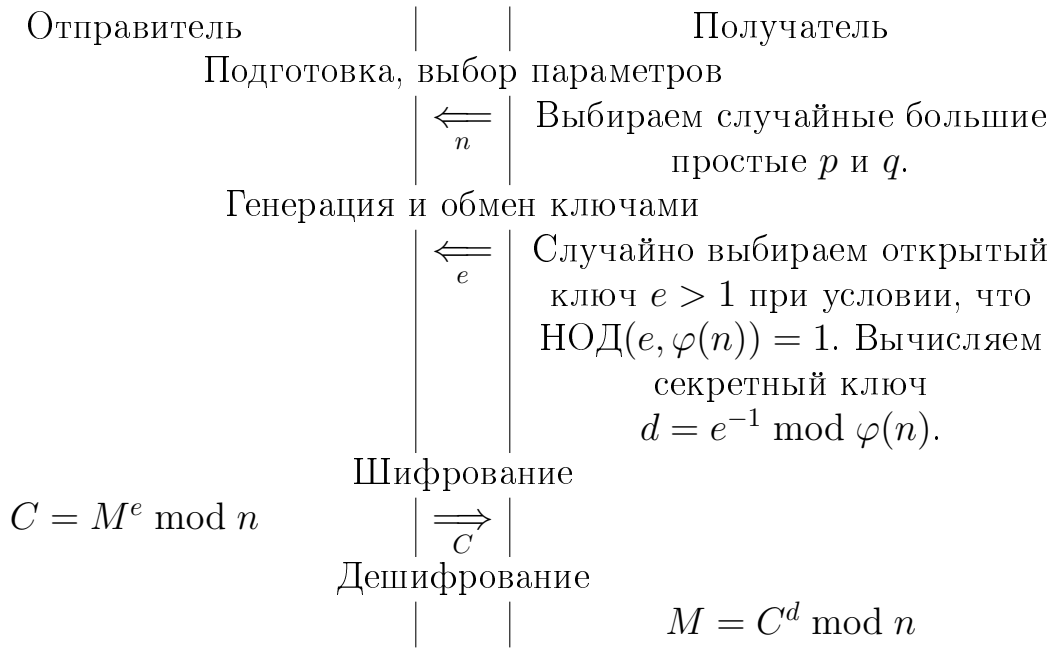


Рис. 7.4. Алгоритм RSA

$C_1 = (3^7) \bmod 33 = 2187 \bmod 33 = 9$;
$C_2 = (1^7) \bmod 33 = 1 \bmod 33 = 1$;
$C_3 = (2^7) \bmod 33 = 128 \bmod 33 = 29$.
Теперь расшифруем данные, используя закрытый ключ $3, 33$.
$M_1 = (9^3) \bmod 33 = 729 \bmod 33 = 3(C)$;
$M_2 = (1^3) \bmod 33 = 1 \bmod 33 = 1(A)$;
$M_3 = (29^3) \bmod 33 = 24389 \bmod 33 = 2(B)$.

Упражнение 62. Может ли в алгоритме RSA Отправитель генерировать p и q?

7.5. Алгоритм шифрования Эль-Гамаля

Алгоритм Эль-Гамаля — криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема (рис. 7.5) была предложена Т. Эль-Гамалем в 1984 году.

Этап — генерация ключей. Генерируется случайное простое число p длины n битов. Выбирается случайный примитивный элемент g поля $\mathbb{Z}_p^*$. Выбирается случайное целое число x такое, что $1 < x < p - 1$. Вычисляется $y = g^x \bmod p$. Открытым ключом является тройка (p, g, y), закрытым ключом — число x.

Этап — шифрование. Сообщение M шифруется следующим образом: выбирается сессионный ключ — случайное целое число k такое, что $1 < k < p - 1$. Вычисляются числа $a = g^k \bmod p$ и $b = y^k M \bmod p$. Пара

чисел (a, b) является шифротекстом. Длина шифротекста в схеме Эль-Гамаля длиннее исходного сообщения M вдвое.

Этап — дешифрование. Зная закрытый ключ x, исходное сообщение можно вычислить из шифротекста (a, b) по формуле: $M = b(a^x)^{-1} \bmod p$. При этом непоскольку $(a^x)^{-1} \equiv g^{-kx} \pmod p$, то $b(a^x)^{-1} \equiv (y^k M)g^{-xk} \equiv (g^{xk}M)g^{-xk} \equiv M \pmod p$. Для практических вычислений больше подходит следующая формула: $M = b(a^x)^{-1} \bmod p = b \cdot a^{(p-1-x)} \bmod p$.

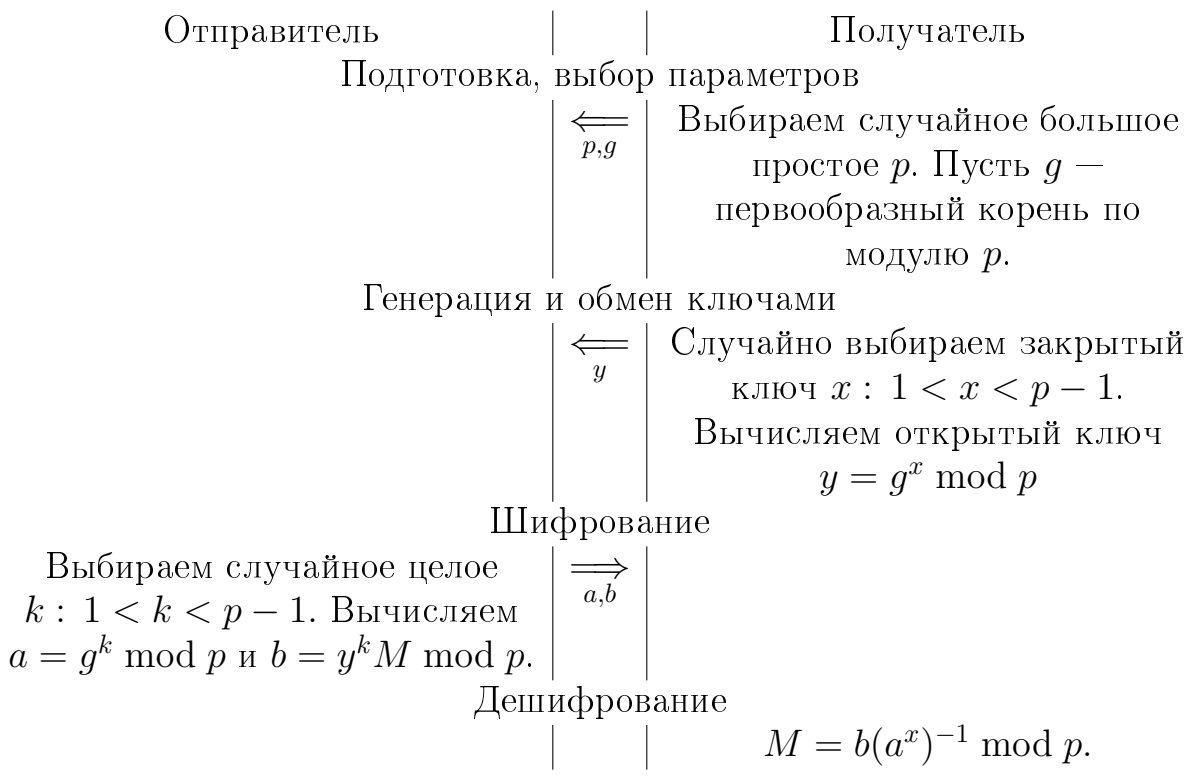


Рис. 7.5. Алгоритм Эль-Гамаля

Пример 51. Произведем генерацию ключей :

Пусть $p = 11, g = 2$. Выберем $x = 8$ — случайное целое число x такое,что $1 < x < p$. Вычислим $y = g^x \bmod p = 2^8 \bmod 11 = 3$. Итак, открытым является тройка $(p, g, y) = (11, 2, 3)$, а закрытым ключом является число $x = 8$.

Шифрование. Допустим что нужно зашифровать сообщение $M = 5$. Выбираем случайное целое число k такое, что $1 < k < (p - 1)$. Пусть $k = 9$. Вычисляем число $a = g^k \bmod p = 2^9 \bmod 11 = 512 \bmod 11 = 6$. Вычисляем число $b = y^k M \bmod p = 3^9 5 \bmod 11 = 19683 \cdot 5 \bmod 11 = 9$. Полученная пара $(a, b) = (6, 9)$ является шифротекстом.

Дешифрование. Вычисляем M по формуле : $M = b(a^x)^{-1} \bmod p = 9(6^8)^{-1} \mod 11 = 5$. Получили исходное сообщение $M = 5$.

Так как в схему Эль-Гамаля вводится случайная величина k, то шифр Эль-Гамаля можно назвать шифром многозначной замены. Из-за слу-

чайности выбора числа k такую схему еще называют схемой вероятностного шифрования. Вероятностный характер шифрования является преимуществом для схемы Эль-Гамаля, так как у схем вероятностного шифрования наблюдается большая стойкость по сравнению со схемами с определенным процессом шифрования. Недостатком схемы шифрования Эль-Гамаля является удвоение длины зашифрованного текста по сравнению с начальным текстом.

Упражнение 63. Как изменится криптостойкость алгоритма Эль-Гамаля, если использовать постоянное значение k?

7.6. Алгоритм шифрования Рабина

Криптосистема Рабина (М. Рабин, 1979 г.) была первой асимметричной криптосистемой, для которой было доказано, что восстановление исходного текста от зашифрованного столь же трудно как факторизация больших чисел. Точнее, она связана с трудностью извлечения квадратного корня по модулю составного числа $n = pq$. Эти две задачи эквивалентны, т. е. — зная простые делители числа N, мы можем извлекать квадратные корни по модулю n, а умея извлекать квадратные корни по модулю n, мы в состоянии разложить n на простые множители.

Этап — генерация ключа. Система Рабина использует и открытый и закрытый ключи. Процесс генерации ключей заключается в следующем. Выбираются два больших простых числа p и q, которые удовлетворяют условию $p \equiv q \equiv 3 \pmod 4$. Такой специальный вид простых чисел сильно ускоряет процедуру извлечения корней по модулю p и q. Тогда $n = p \cdot q$ и n — открытый ключ**.** Числа p и q — закрытый ключ.

Этап — шифрование. Для шифрования сообщения m нужно вычислить

$$c = m^2 \bmod n.$$

Таким образом, шифрование состоит из операции умножения по модулю n, что обеспечивает более высокую скорость шифрования, чем в RSA, даже если в последней выбирают небольшую шифрующую экспоненту.

Этап — дешифрование. Сначала, используя расширенный алгоритм Евклида, из уравнения $y_p \cdot p + y_q \cdot q = 1$ находим числа y_p и y_q. Далее, используя китайскую теорему об остатках, можно вычислить числа

$$\begin{array}{rcl} r & = & (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \bmod n \\ -r & = & n - r \\ s & = & (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \bmod n \\ -s & = & n - s \end{array} \qquad (7.1)$$

Один из этих корней $r, -r, s, -s$ является истинным открытым текстом m .

Пример 52. Пусть $p = 7$ и $q = 11$, тогда $n = 77$. Открытый ключ 77, публикуется для всеобщего обозрения, с помощью него шифруются сообщения. Закрытые ключи 7 и 11, остаются известны только владельцу, и с помощью их расшифровываются сообщения. Пусть исходным текстом является $m = 20$. Тогда зашифрованным текстом будет:

$$c = m^2 \bmod 77 = 400 \bmod 77 = 15.$$

Для дешифрования решим уравнение $y_p \cdot 7 + y_q \cdot 11 = 1$, получим $y_p = -3$, $y_q = 2$. Вычислим m_p, m_q, r: $m_p = m \bmod 7 = 20 \bmod \{7\} = 6$, $m_q = m \bmod 11 = 20 \bmod 11 = 9$.

Согласно (7.1) имеем $r = (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \bmod n = (-3 \cdot 7 \cdot 9 + 2 \cdot 11 \cdot 6) \bmod 77 = -57 \bmod 77 = 20$, следовательно, $-r = 77 - 20 = 57$. Вычислим s. Аналогично $s = (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \bmod n = (-3 \cdot 7 \cdot 9 - 2 \cdot 11 \cdot 6) \bmod 77 = -57 \bmod 77 = 64$, следовательно, $-s = 77 - 64 = 13$. В результате расшифровки получаем: $m \in \{64, \mathbf{20}, 13, 57\}$. Видим, что один из корней является исходным текстом m.

Упражнение 64. Выведите формулу (7.1).

7.7. Алгоритм шифрования на основе эллиптических кривых

Для шифрования с помощью эллиптической кривой может быть применен следующий алгоритм. Пользователи A и B выбирают некоторую эллиптическую кривую $E_p(a, b)$ и точку G на ней (порядок точки G — большое простое число), а также секретные ключи — целые числа n_A и n_B соответственно. Открытые ключи $P_A = n_A G$ и $P_B = n_B G$ могут быть свободно распространены между пользователями. Исходному алфавиту сопоставляются некоторые точки кривой. Для шифрования некоторой точки кривой P_m пользователь A генерирует случайное целое k и формирует шифротект, состоящий из пары точек, по формуле $C = (kG, P_m + kP_B)$. Для дешифрования шифротекта C пользователь B вычисляет $P_m + kP_B - n_B kG = P_m + kn_B G - n_B kG = P_m$ и получает исходный текст P_m (рис. 7.6).

Пример 53. Рассмотрим эллиптическую кривую $E_{11}(-2, 5)$, т. е. кривая описывается соотношением $y^2 \equiv x^3 - 2x + 5 \pmod{11}$. Пусть выбрана точка $G = (7, 2)$. Личный ключ участника A равен $n_A = 2$ и ему соответствует точка $P_A = 2G = (0, 7)$. Личным ключом пользователя B является число $n_B = 6$ и ему соответствует точка $P_B = 6G = (7, 9)$. Для шифрования сообщения будем использовать следующий алфавит: точка (0,7) будет соответствовать нулевому биту, а точка (1,2) — единичному.

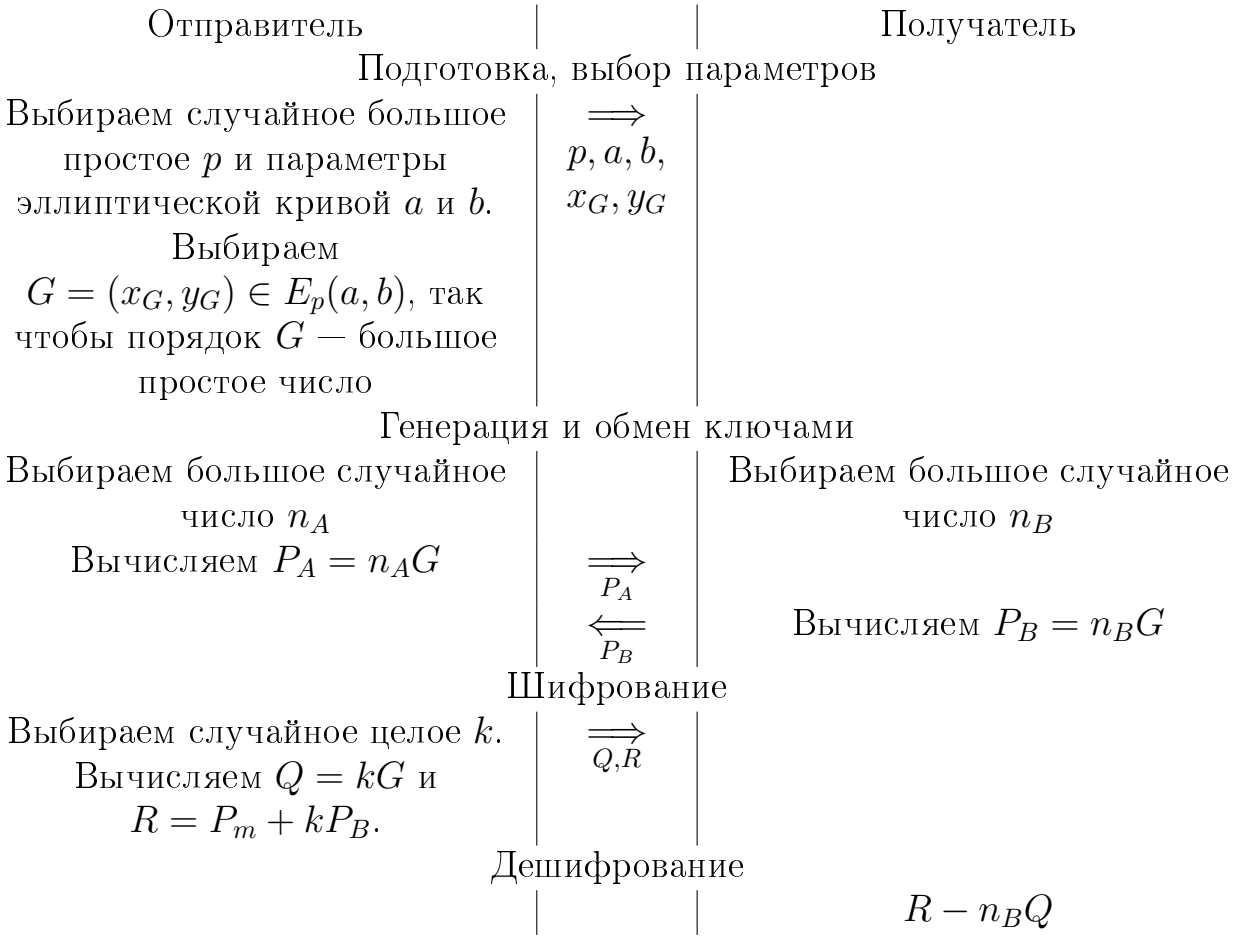


Рис. 7.6. Шифрование на основе эллиптических кривых

Мы хотим передать текст, состоящий из последовательности бит: “0, 1”. Для шифрования первого бита выберем случайное целое $k = 4$ и сформируем шифротекст $C_1 = (4G, (0, 7) + 4(7, 9)) = ((9, 10), (0, 4))$, аналогично выберем случайное $k = 2$ для шифрования второго бита и сформируем шифротекст $C_2 = (2G, (1, 2) + 2(7, 9)) = ((0, 7), (3, 2))$. Для дешифрования первого бита вычислим $M_1 = (0, 4) - 6(9, 10) = (0, 4) - (9, 1) = (0, 4) + (9, 10) = (0, 7)$, а точка (0,7) соответствует нулевому биту. Аналогично дешифруем C_2. $M_2 = (3, 2) - 6(0, 7) = (3, 2) - (0, 4) = (3, 2) + (0, 7) = (1, 2)$, что соответствует единичному биту.

Упражнение 65. Необходимо ли, чтобы точка P_m входила в подгруппу порожденную точкой G?

7.8. Электронная цифровая подпись на основе эллиптических кривых

Использование некоторых алгоритмов ассиметричного шифрования позволяет реализовать электронную цифровую подпись, которая обеспечивает такие важные сервисы безопасности как целостность, неотказуе-

мость, авторство. Схема одного из возможных применений электронной подписи представлена на рис. 7.7.

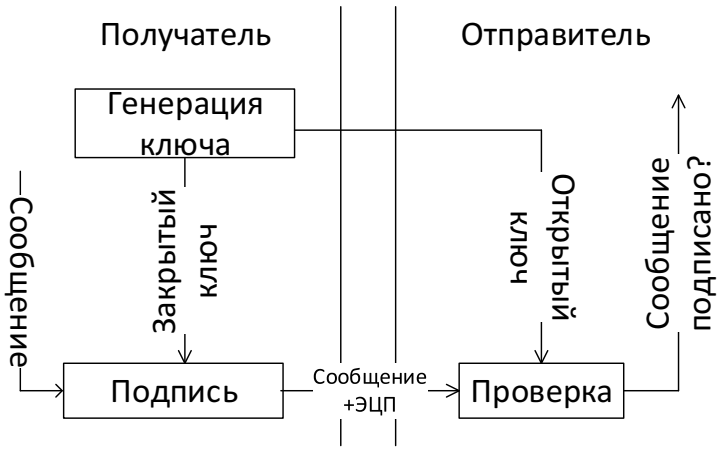


Рис. 7.7. Схема электронной цифровой подписи

Реализация электронной цифровой подписи описана в ГОСТ 34.10-2012 (рис. 7.8). Для генерации подписи стороны выбирают эллиптическую кривую $E_p(a,b)$, точку $G \in E_p(a,b)$ такую, что порядок G — большое простое число q. Дальше каждый пользователь генерирует себе секретный ключ: случайное целое число $d (0 < d < q)$ и точку $Q = d \cdot G$. Кроме того, к выбранным параметрам предъявляются следующие ограничения:

1. Должно выполняться сравнение $p^t \neq 1 \pmod q$ для $t = \overline{1, B}$, где $B = 31$, если $2^{254} < p < 2^{256}$, и B=131, если $2^{508} < p < 2^{512}$.

2. Порядок группы точек $E_p(a,b)$ не должен совпадать с p $(q < p)$.

3. Инвариант кривой должен удовлетворять соотношению $J(E) \neq \{0, 1728\}$.

Для подписи стадарт предусматривает использовать хэш сообщения a. Формирование цифровой подписи состоит из следующих этапов.

1. Вычислить $e = a \bmod q$. Если $e = 0$, то $e \leftarrow 1$.

2. Сгенерировать случайное число k $(0 < k < q)$.

3. Вычислить точку на эллиптической кривой

$$C = kG \tag{7.2}$$

и определить $r = x_C \bmod q$. Если $r = 0$, то вернуть на предыдущий шаг.

4. Вычислить

$$s = (rd + ke) \bmod q,$$

где d – секретный ключ. Если $s = 0$, то вернуться к Шагу 3.

5. Пара (r, s) являются цифровой подписью сообщения a.

Проверка цифровой подписи заключается в выполнении следующих шагов:

1. Аналогично вычисляем хэш сообщения a.

2. Вычислить $e = a \bmod q$. Если $e = 0$, то $e \leftarrow 1$.

3. Вычислить $v = e^{-1} \bmod q$.

4. Вычислить $z_1 = sv \bmod q$ и $z_2 = -rv \bmod q$.

5. Вычислить точку эллиптической кривой

$$C = z_1 G + z_2 Q \tag{7.3}$$

и определить $R = x_C \bmod q$.

6. Если $R = r$, то подпись верна, иначе нет.

Покажем, что подпись должна работать корректно:

$$C = z_1 G + z_2 Q = svG - rvQ = se^{-1}G - re^{-1}Q =$$
$$= (rd + ke)\, e^{-1}G - re^{-1}dG = rde^{-1}G + kee^{-1}G - re^{-1}dG =$$
$$= kee^{-1}G = kG = C,$$

здесь первое равенство из (7.3), а последнее из (7.2).

Отправитель		Получатель
Подготовка, выбор параметров		
Выбираем случайное большое простое p и параметры эллиптической кривой a и b. Выбираем $G = (x_G, y_G) \in E_p(a,b)$, так чтобы порядок G — большое простое число	$\Longrightarrow$ p, a, b, x_G, y_G	
Генерация и обмен ключами		
Выбираем большое случайное число d. Вычисляем $Q = dG$	$\underset{Q}{\Longrightarrow}$	
Подписывание		
Пусть e — хэш сообщения. Выбираем случайное целое k. Вычисляем $r = x_C \bmod q$, $s = (rd + ke) \bmod q$.	$\underset{r,s}{\Longrightarrow}$	
Проверка подписи		
		Пусть e — хэш сообщения. Вычислить $v = e^{-1} \bmod q$, $z_1 = sv \bmod q$, $z_2 = -rv \bmod q$, $C = z_1 G + z_2 Q$, $R = x_C \bmod q$. Подпись верна $\Longleftrightarrow R = r$.

Рис. 7.8. ЭЦП на основе эллиптических кривых

Пример 54. Рассмотрим пример применения ЭЦП. Выберем поле $E_{11}(-2, 5)$. Пусть точка $G = (7, 2)$, тогда порядок подгруппы $q = 7$. И

пусть секретный ключ $d = 2$, тогда $Q = (0, 7)$. Предположим, что хэш сообщения выражается как $e = 5$. Выберем случайно $k = 3$. Вычисляем точку на эллиптической кривой $C = kG = (9, 1)$. Найдем $r = 9 \bmod 7 = 2$. Вычислим s. $s = (rd + ke) \bmod q = (2 \cdot 2 + 3 \cdot 5) \bmod 7 = 5$.Цифровая подпись готова - это пара чисел $(2, 5)$.

Проверим цифровую подпись. Вычислим $v = e^{-1} \bmod q = 5^{-1} \bmod 7 = 3$. Найдем z_1 и z_2. $z_1 = sv \bmod q = 5 \cdot 3 \bmod 7 = 1$, а $z_2 = -rv \bmod q = -2 \cdot 3 \bmod 7 = 1$. Найдем точку C. $C = G + Q = (7, 2) + (9, 1) = (9, 10)$. Вычислим $R = 9 \bmod 7 = 2$. Т. к. $R = r$, то подпись верна.

Упражнение 66. Что произойдет с цифровой подписью при $e = 0$?

8. ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Все задания предполагают выполнение без использования калькуляторов.

1. Расшифровать шифротекст “**Э Д А Ь Е**”, если известны следующие пары сообщения-шифротекст: просто — “О Э Е А О Ф”, талант — ”Ь Я Е Е Т Ь”, дата — “Ь Ь Е Ц”, команда — “Я Ь О С Ц Ь П”, лидерство — ”Д Е Т А Ц Г О Щ Ф”,развитие — ”Ь Д А Щ Е Г Д У”.

2. Пользуясь методом повторного возведения в квадрат, вычислить:

1. $23^{36} \bmod 78$;
2. $38^{75} \bmod 103$;
3. $45^{63} \bmod 59$;
4. $30^{30} \bmod 65$.
5. $26^{15} \bmod 13$;
6. $5^{23} \bmod 12$;
7. $23^{8} \bmod 21$;
8. $27^{15} \bmod 14$;
9. $8^{14} \bmod 16$;
10. $18^{24} \bmod 10$;
11. $13^{6} \bmod 20$;
12. $14^{5} \bmod 2$;
13. $25^{21} \bmod 29$;
14. $12^{9} \bmod 24$;
15. $23^{22} \bmod 14$;
16. $24^{18} \bmod 23$;
17. $20^{24} \bmod 18$;
18. $22^{8} \bmod 18$;
19. $4^{3} \bmod 22$;
20. $25^{20} \bmod 15$;
21. $26^{13} \bmod 29$;
22. $26^{9} \bmod 27$;
23. $21^{7} \bmod 17$;
24. $26^{15} \bmod 13$.

3. Найдите значения:

1. $\varphi(29)$;
2. $\varphi(32)$;
3. $\varphi(80)$;
4. $\varphi(100)$;
5. $\varphi(101)$.
6. $\varphi(7)$;
7. $\varphi(13)$;
8. $\varphi(16)$;
9. $\varphi(40)$;
10. $\varphi(15)$;
11. $\varphi(14)$;
12. $\varphi(18)$;
13. $\varphi(20)$;
14. $\varphi(7)$;
15. $\varphi(24)$;
16. $\varphi(28)$;
17. $\varphi(19)$;
18. $\varphi(33)$.

4. Найдите результаты с использованием малой теоремы Ферма

1. $5^{15} \bmod 13$;
2. $15^{18} \bmod 17$;
3. $456^{17} \bmod 17$;
4. $5^{-1} \bmod 13$;
5. $15^{-1} \bmod 17$;
6. $27^{-1} \bmod 41$;
7. $70^{-1} \bmod 101$.
8. $6^{-1} \bmod 11$;
9. $7^{-1} \bmod 23$;
10. $8^{-1} \bmod 47$;
11. $9^{-1} \bmod 19$;
12. $10^{-1} \bmod 7$;
13. $11^{-1} \bmod 5$;

14. $12^{-1} \bmod 41$;
15. $13^{-1} \bmod 43$;
16. $14^{-1} \bmod 23$;
17. $15^{-1} \bmod 29$;
18. $16^{-1} \bmod 31$;
19. $17^{-1} \bmod 17$;
20. $18^{-1} \bmod 19$.

5. Покажите, что $2^{24}-1$ и $2^{16}-1$ — составные числа.

6. Будет ли циклической мультипликативная группа $\mathbb{Z}_{24}^*$?

7. Сколько порождающих элементов в мультипликативной группе $\mathbb{Z}_{23}^*$?

8. Для группы $G = \langle \mathbb{Z}_{19}^*, * \rangle$ найдите порядок группы и порядок каждого элемента в группе.

9. Решить линейные сравнения:

1. $2x \equiv 3 \pmod 5$;
2. $3x \equiv 4 \pmod 7$;
3. $7x \equiv 10 \pmod{11}$;
4. $12x \equiv 7 \pmod{13}$;
5. $7x \equiv 11 \pmod{15}$;
6. $5x \equiv 3 \pmod{17}$;
7. $3x \equiv 5 \pmod{11}$;
8. $9x \equiv 2 \pmod{14}$;
9. $10x \equiv 15 \pmod{25}$;
10. $9x \equiv 12 \pmod{21}$;
11. $28x \equiv 40 \pmod{44}$;
12. $24x \equiv 14 \pmod{26}$;
13. $21x \equiv 33 \pmod{45}$;
14. $30x \equiv 18 \pmod{102}$;
15. $21x \equiv 35 \pmod{77}$;
16. $18x \equiv 22 \pmod{11}$;
17. $15x \equiv 16 \pmod{19}$;
18. $12x \equiv 18 \pmod{12}$;
19. $8x \equiv 9 \pmod{10}$;
20. $30x \equiv 29 \pmod 3$;
21. $12x \equiv 6 \pmod{25}$;
22. $23x \equiv 26 \pmod{24}$;
23. $9x \equiv 5 \pmod 6$;
24. $17x \equiv 7 \pmod 1$;
25. $25x \equiv 11 \pmod{20}$;
26. $28x \equiv 5 \pmod 9$;
27. $15x \equiv 25 \pmod 6$;
28. $23x \equiv 15 \pmod{27}$;
29. $19x \equiv 25 \pmod{15}$;
30. $27x \equiv 20 \pmod{22}$;
31. $17x \equiv 8 \pmod{10}$;
32. $6x \equiv 8 \pmod 7$;
33. $26x \equiv 17 \pmod{26}$;
34. $7x \equiv 26 \pmod{21}$;
35. $10x \equiv 21 \pmod{14}$;
36. $3x \equiv 12 \pmod{22}$;
37. $28x \equiv 19 \pmod{22}$;
38. $27x \equiv 24 \pmod 6$;
39. $8x \equiv 23 \pmod{26}$;
40. $6x \equiv 24 \pmod{22}$;
41. $14x \equiv 6 \pmod{25}$.

10. Решить системы сравнений:

1. $\begin{cases} x \equiv 3 \pmod{11}; \\ x \equiv 5 \pmod 7; \end{cases}$
2. $\begin{cases} x \equiv 6 \pmod 7; \\ x \equiv 2 \pmod{13}; \end{cases}$
3. $\begin{cases} x \equiv 3 \pmod{17}; \\ 3x \equiv 6 \pmod 9; \end{cases}$
4. $\begin{cases} x \equiv 7 \pmod{11}; \\ x \equiv 3 \pmod{10}; \\ x \equiv 2 \pmod 3; \end{cases}$
5. $\begin{cases} 7x \equiv 10 \pmod{11}; \\ 12x \equiv 7 \pmod{13}; \\ 7x \equiv 11 \pmod{15}; \end{cases}$
6. $\begin{cases} x \equiv 13 \pmod{16}; \\ x \equiv 3 \pmod{10}; \\ x \equiv 9 \pmod{14}; \end{cases}$
7. $\begin{cases} x \equiv 4 \pmod{15}; \\ x \equiv 1 \pmod{12}; \\ x \equiv 7 \pmod{14}. \end{cases}$

8. $\begin{cases} 11x \equiv 18 \pmod{11}; \\ 22x \equiv 24 \pmod{16}; \\ 30x \equiv 26 \pmod{15}; \end{cases}$

9. $\begin{cases} 29x \equiv 12 \pmod{22}; \\ 23x \equiv 25 \pmod{30}; \end{cases}$

10. $\begin{cases} 8x \equiv 29 \pmod{5}; \\ 10x \equiv 26 \pmod{6}; \end{cases}$

11. $\begin{cases} 27x \equiv 29 \pmod{6}; \\ 14x \equiv 26 \pmod{27}; \\ 21x \equiv 16 \pmod{5}; \\ 5x \equiv 17 \pmod{21}; \end{cases}$

12. $\begin{cases} 23x \equiv 25 \pmod{23}; \\ 25x \equiv 29 \pmod{16}; \\ 22x \equiv 19 \pmod{28}; \\ 15x \equiv 7 \pmod{28}; \end{cases}$

13. $\begin{cases} 13x \equiv 5 \pmod{28}; \\ 5x \equiv 8 \pmod{23}; \end{cases}$

11. Найти числа, которые при делении на 13, 5 и 12 дают в остатке 5, 1 и 7.

12. Найти числа, которые при делении на 7, 11 и 13 дают в остатке 3, 2 и 5.

13. Найти числа, которые при делении на 7, 11 и 17 дают в остатке 3, 5 и 13.

14. Вычислить символы Лежандра:

1. $L(1801, 8191)$;
2. $L(7411, 9283)$;
3. $L(19, 31)$;
4. $L(97, 101)$;
5. $L(219, 383)$;
6. $L(312, 199)$;
7. $L(5, 160465489)$;
8. $L(134, 139)$;
9. $L(208, 727)$;
10. $L(3083, 3911)$;
11. $L(11, 47)$;
12. $L(76, 97)$;
13. $L(11, 53)$;
14. $L(98, 61)$;
15. $L(85, 17)$;
16. $L(73, 101)$;
17. $L(15, 59)$;
18. $L(16, 51)$;
19. $L(77, 19)$;
20. $L(91, 97)$;
21. $L(79, 31)$;
22. $L(54, 71)$;
23. $L(88, 53)$;
24. $L(53, 29)$.

15. Найти наименьший положительный невычет n по модулю p:

1. $p = 887$;
2. $p = 2081$;
3. $p = 2357$;
4. $p = 1931$.

16. Используя квадратичные вычеты, решите следующие сравнения:

1. $x^2 \equiv 4 \pmod{7}$;
2. $x^2 \equiv 5 \pmod{11}$;
3. $x^2 \equiv 7 \pmod{13}$;
4. $x^2 \equiv 12 \pmod{17}$;
5. $x^2 \equiv 4 \pmod{14}$;
6. $x^2 \equiv 5 \pmod{10}$;
7. $x^2 \equiv 7 \pmod{33}$;
8. $x^2 \equiv 12 \pmod{34}$.
9. $x^2 \equiv 12 \pmod{91}$;
10. $x^2 \equiv 78 \pmod{83}$;
11. $x^2 \equiv 75 \pmod{59}$;
12. $x^2 \equiv 54 \pmod{99}$;
13. $x^2 \equiv 35 \pmod{39}$;
14. $x^2 \equiv 22 \pmod{87}$;
15. $x^2 \equiv 76 \pmod{79}$;
16. $x^2 \equiv 73 \pmod{71}$;
17. $x^2 \equiv 62 \pmod{67}$;

18. $x^2 \equiv 78 \pmod{79}$;
19. $x^2 \equiv 62 \pmod{67}$;
20. $x^2 \equiv 46 \pmod{67}$;
21. $x^2 \equiv 5 \pmod{19}$;
22. $x^2 \equiv 36 \pmod{83}$.

17. С помощью алгоритма Полига-Хеллмана вычислить $\log_2 28$ в поле $\mathbb{Z}_{37}^*$.

18. С помощью алгоритма Полига-Хеллмана решить сравнение $3^x \equiv 15 \pmod{17}$.

19. Вычислить дискретный логарифм элемента 25 в поле $\mathbb{Z}_{41}^*$ относительно порождающего элемента $g = 7$.

20. Используя лишь калькулятор, найти число x, удовлетворяющее сравнению $3^x \equiv 5 \pmod{p}$, где $p-1 = 2 \cdot 3 \cdot 11$ или $p-1 = 2 \cdot 3 \cdot 101$.

21. Сколько решений уравнения $x^p = 1$ имеется в простом конечном поле $\mathbb{Z}_p^*$?

22. Найти первообразные корни по модулю:

1. 11
2. 17
3. 23
4. 29
5. 7
6. 8
7. 12
8. 19
9. 31

23. Отправитель, используя открытый ключ RSA получателя ($e = 17$, $n = 19519$), чтобы передать сообщение из четырех символов Получателю, применяющему схему кодирования $A \leftrightarrow 0, B \leftrightarrow 1, ..., Z \leftrightarrow 25$ по каждому символу отдельно. Противник перехватывает зашифрованный текст (6625 0 2968 17683) и расшифровывает сообщение, не разлагая на множители модуль. Найдите исходный текст, объясните почему удалось взломать зашифрованный текст.

24. Пусть $n = pq$ модуль система RSA. Пусть e — открытый ключ шифрования. Предположим, что известен порядок элемента e в группе $\mathbb{Z}_{\varphi(n)}^*$ и этот порядок равен k. Как можно дешифровать сообщение $m^e \pmod{n}$ и даже вычислить изначально секретный ключ d?

25. Предположим нам удалось перехватить сообщение c, зашифрованное алгоритмом RSA с открытыми параметрами (e, n). Получатель готов подписать нам любое число, которое он раньше не видел. Что необходимо подписать у получателя, чтобы прочитать сообщение c?

26. Два Получателя используют следующие открытые ключи (e_1, n) и (e_2, n). Каждому Получателю было послано неизвестное сообщение m, а шифрограммы стали доступны Злоумышленнику. Как узнать m?

27. Перечислите все точки эллиптической кривой $E_7(1, 4)$. Каков порядок группы $E_7(1, 4)$?

28. Постройте таблицу Кэли для группы точек кривой $E_7(1, 4)$.

29. Укажите порядок всех точек в группе $E_7(1, 4)$.

30. Зашифруйте и расшифруйте сообщение “01”, используя кривую $E_7(1, 4)$. Остальные параметры выберите самостоятельно.

9. ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторные работы выполняются в бригадах. В каждую бригаду могут входить не более двух студентов. Сдача лабораторной работы в срок, означает сдачу на следующем занятии после занятия, на котором лабораторная работа была выдана. Сдача лабораторной работы возможна и на последующих занятиях, но с обязательным устным ответом на дополнительные вопросы по теоретической части лабораторной работы.

9.1. Криптоанализ моноалфавитного шифра

Одним из простейших шифров подстановки является моноалфавитный шифр. Рассмотрим исходное сообщение M, пусть оно составлено из символов алфавита $\mathcal{A}$. Тогда моноалфавитным шифром называется отображение $\mathcal{A} \to \mathcal{A}$. Ключом для этого шифра является отображение $\mathcal{A} \to \mathcal{A}$.

Особенностью моноалфавитного шифра является сохранение частотных характеристик символов исходного текста в шифротексте. Этим можно воспользоваться для проведения криптоатаки. Предположим, что исходный текст является текстом на естественном языке. В естественных языках частоты встречаемости символов очень неравномерны. Поскольку моноалфавитный шифр не изменяет частот встреч символов, то сопоставляя частотную таблицу для эталонного текста и шифротекста у нас есть возможность сделать предположения о ключе шифра.

Необходимо отметить, что если объем эталонного текста для нас может быть не ограничен, то объем шифротекста всегда ограничен. Поэтому при построении частотной таблицы у нас неизбежно возникают погрешности в вычислении оценок частот встреч букв. При больших частотах встречи букв это будет менее значимо, поскольку статистики будет хватать для верного упорядочивания букв шифротекста, на редковстречаемых символах – ошибка может значительно искажать картину. Поэтому расшифровав самые часто встречаемые буквы с помощью частотного анализа необходимо использовать дополнительную информацию для криптоанализа остального текста.

Задание на лабораторную работу

С помощью метода частного криптоанализа восстановить исходный текст шифрограммы, зашифрованной моноалафитным шифром.

Порядок действий

1. Выбрать шифротекст для своего варианта (номера бригады).
2. Сформировать частотную таблицу для эталонного текста на русском языке.
3. Сформировать частотную таблицу для шифротекста.
4. Сопоставить частотные таблицы расшифровать пробел в исходном тексте.
5. Перебирая различные варианты расшифровки частовстречаемых символов в исходном тексте на основе частотных таблиц провести частичную дешифровку специфичных слов русского языка (короткие предлоги, местоимения, окончания слов и т. д.).
6. Подбором расшифровать оставшиеся редковстречаемые символы в шифротексте.
7. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Шифротекст.
5. Сопоставленные частотные таблицы для эталонного текста и шифротекста.
6. Ключ моноалавитного шифра.
7. Расшифрованный исходные текст.
8. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Сколько всевозможных ключей для моноалфавитного шифра с алфавитом $\mathcal{A}$?
2. Насколько неравномерны частоты встреч символов на искусственном языке (например, программный код на языке C/C++)?
3. В чем заключается сложность оценивания частоты встречи редковстречаемых символов в шифротексте?
4. Приведите примеры моноалфавитных шифров.
5. Выделите самый криптостойкий из моноалфавитных шифров.
6. Есть ли заведомо слабые ключи в моноалфавитных шифрах?

9.2. Криптоанализ полиалфавитного шифра

Поскольку моноалфавитные шифры оказались нестойкими к частотному криптоанализу, следующим закономерным этапом развития шифров стало повышение их стойкости к частотному криптоанализу. Рассмотрим полиалфавитный шифр – это такой шифр, в котором каждый символ исходного текста шифруется с помощью различного мноалфавитного шифра. Частотный анализ полиалфавитного шифра не даст эффекта поскольку все частоты символов оказываются примерно выровненными.

Криптоанализ полиалфавитного шифра можно провести следующим образом. Первая задача – определение длины ключа, после этого задача разбивается на криптоанализ нескольких моноалфавитных шифров. Для определения длины ключа ключа можно воспользоваться двумя взаимодополняющими методами: методом Казиски и методом индекса соответствия.

Метод Казиски основан на следующих соображениях: если два одинаковых отрезка открытого текста получаются один из другого сдвигом на величину, кратную длине ключевого слова, то они при шифровании перейдут в одинаковые отрезки шифротекста. Эти аргументы используются следующим образом: если в шифротексте имеются два фрагмента длины три или больше, то весьма вероятно, что они соответствуют одинаковым отрезкам открытого текста. Находим расстояние между стартовыми позициями этих отрезков. Если мы найдем несколько таких расстояний L_1, L_2, ..., то можно предположить, что искомое число L (длина ключа) является делителем каждого из этих чисел и, следовательно, делителем их наибольшего общего делителя. Дальнейшие аргументы для нахождения числа L могут быть получены с помощью индексов совпадения.

Определение 29. Пусть s — строка, составленная из букв некоторого алфавита. *Индексом соответствия* строки s называется вероятность того, что две случайно выбранные буквы этой строки являются одинаковыми.

Эмпирически индекс соответствия вычисляется как

$$I(s) = \sum_{i=\overline{1,k}} f_i \frac{f_i - 1}{n(n-1)},$$

где f_i — количество встречи i-ой буквы в строке s, k — общее количество букв в алфавите, n — длина строки. Важным свойством является то, что индекс совпадения для строки не меняется при шифровании этой строки моноалфавитным шифром. Для случайной строки символов алфавита

индекс соответствия примерно равен $I(s) \approx \frac{1}{n}$ (для случайного набора русских букв примерно 0.03). Для текстов на русском языке индекс соответствия равен примерно 0,0553. Соответственно, перебирая длину ключа, для каждой длины мы получаем m моноалфавитных шифров. Для каждого моноалфавитного шифра вычисляем индекс соответствия, в качестве оценки длины ключа выбираем среднее от всех индексов соответствия. В качестве длины ключа выбираем длину, у которой эмпирический индекс близок к ожидаемому.

Задание на лабораторную работу

С помощью метода Казиски и индекса соответствия определить длину ключа и с помощью частного криптоанализа восстановить исходный текст шифрограммы, зашифрованной полиалафитным шифром — шифром Виженера.

Порядок действий

1. Выбрать шифротекст для своего варианта (номера бригады).
2. Вычислить возможные длины ключа методом Казиски.
3. Вычислить возможные длины ключа методом индекса соответствия.
4. Определить возможную длину ключа.
5. Для каждого блока символов, зашифрованной моноалфавитным шифром, с помощью частотного криптоанализа определить пробел в исходном тексте и определить по пробелу используемый сдвиг в каждом блоке.
6. Расшифровать шифротекст с использованием полученных на предыдущем этапе сдвигов.
7. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Шифротекст.
5. Список повторяющихся подстрок и множество простых множителей, найденных методом Казиски.
6. Рассчитанные значения индекса соответствия для длины ключа от 2 до 20.
7. Найденные сдвиги для каждого блока символов.
8. Расшифрованный исходные текст.

9. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Сколько всевозможных ключей для полиалфавитного шифра с алфавитом A и длиной ключа N?

2. Что делать, если длина ключа равно длине шифротекста?

3. Какой объем текста необходим для криптоанализа полиалфавитного шифра по сравнению с моноалафитным? (Шифр Виженера или общий случай)

9.3. Криптоанализ методом вероятных слов

Одним из методов криптоанализа является метод вероятных слов. Этот метод заключается в следующем. Пусть для шифрования использован шифр гаммирования, который описывается соотношениями:

$$\begin{aligned} C_i &= M_i \oplus K_{i \bmod L}, \\ M_i &= C_i \oplus K_{i \bmod L}, \end{aligned} \tag{9.1}$$

где последовательность K называется гаммой. Предположим, что нам известен небольшой участок исходного текста. Тогда с учетом (9.1) получим, что если выполнить операцию сложения по модулю 2 между шифротекстом и известным текстом, то можно восстановить участок гаммы. Если для шифирования использована короткая гамма, которая циклически повторяется, то появляется возможность вскрыть текст полностью.

В качестве вероятностных слов для русского текста возможны следующие варианты: «_ОН_», «_МЫ_», «ННЫЕ_», и т.д. Алгоритм взлома шифротекста с помощью метода вероятностных слов:

1. Выбрать вероятностное слово длины большей, чем длина гаммы.

2. Для каждого сдвига вероятностного слова выполнить гаммирование шифротекста и вероятностного слова.

3. Если цикличность подтвердилась, то необходимо проверить гамму и расшифровать весь шифротекст.

Задание на лабораторную работу

С помощью метода вероятных слов восстановить исходный текст шифрограммы, зашифрованной шифром гаммирования.

Порядок действий

1. Выбрать шифротекст для своего варианта (номера бригады).

2. Подготовить программу для проверки вероятных слов.

3. Проверить вероятные слова для нахождения гаммы.
4. Расшифровать шифротекст.
5. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Шифротекст.
5. Проверенные вероятностные слова с результатом проверки.
6. Расшифрованный исходные текст.
7. Программа проверки вероятностных слов.
8. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Каковы условия применения метода вероятных слов (исходный текст, алгоритм дешифрования)?
2. Какова трудоемкость метода вероятных слов?
3. Как можно полностью автоматизировать метод вероятностных слов?
4. Как можно усложнить шифр, чтобы противостоять методу вероятностных слов?

9.4. Криптоанализ методом полного перебора

Одним из самых распространенных методов криптоанализа является метод полного перебора. Суть этого метода заключается в следующем. Рассмотрим схему дешифрования

$$M = F(C, K),$$

где C — известный шифротекст, M — исходный текст, который требуется найти, F — известное функциональное преобразование и K — единственное неизвестное значение, ключ. Единственное, что известно про ключ, это то, что ключ принадлежит некоторому конечному множеству. Поскольку множество конечное, можно перебрать каждый его элемент и расшифровать исходный текст.

Кроме того, что современные алгоритмы шифрования практически не могут быть взломаны «в лоб» алгоритмом полного перебора, часто возникает проблема, связанная с автоматическим распознаванием нужного исходного текста. Например, если известно, что исходный текст на

естественном языке, то может потребоваться проверить миллионы строк — являются ли они строками на естественном языке. Один из стандартных подходов к решению этой задачи — использование таблиц биграмм.

Таблица биграмм – это таблица, каждой строке и каждому столбцу сопоставлен некий символ алфавита, а в ячейке с индексами $[i, j]$ хранится вероятность того, что после символа с индексом i следует символ с индексом j. Это позволяет вычислить вероятность того, что вся строка принадлежит русскому языку по формуле

$$P(s) = \prod_{i=1}^{Length(s)-1} p_{s(i),s(i+1)},$$

где $p_{s(i),s(i+1)}$ — вероятность, что после символа $s(i)$ следует символ $s(i+1)$. Эта формула удобна для теоретического анализа, однако, при практической реализации при большом количестве умножений мантисса переполняется и удобнее вместо произведения использовать, например, сумму.

Задание на лабораторную работу

С помощью метода полного перебора восстановить исходный текст шифрограммы, зашифрованной шифром Виженера с длинной ключа 3.

Порядок действий

1. Выбрать шифротекст для своего варианта (номера бригады).
2. Сформировать таблицу биграмм для эталонного текста на русском языке.
3. Разработать программу для полного перебора ключей длинны 3 в шифре Виженера и оценки вероятности для каждого расшифрованного исходного текста.
4. Найти 10 расшифрованных строк с самой большой оценкой вероятности.
5. Расшифровать шифротекст.
6. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Шифротекст.
5. Таблица биграмм.

6. 10 расшифрованных шифротекстов с максимальной оценкой вероятности.

7. Расшифрованный исходные текст.

8. Ключ шифра Виженера.

9. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Сколько всевозможных ключей для шифра Виженера с алфавитом A?

2. Каковы особенности таблицы биграмм (зависимы ли последовательные символы, насколько)?

3. Каковы предельные возможности полного перебора?

4. Какие есть еще способы проверки расшифрованных исходных текстов?

5. Какова ресурсоемкость построенного алгоритма?

6. Какой метод криптоанализа лучше из рассмотренных?

7. Является ли какой-то из ключей более надежным в смысле алгоритма полного перебора?

9.5. Стеганография

Стеганография это наука, изучающая способы сокрытия каналов связи. Если в криптографии говорят о сокрытии текста сообщения, то в стеганографии скрывают сам факт наличия этого сообщения. В стеганографии помимо сообщения, которое мы хотим передать нам требуется некий контейнер. Контейнер — это любая информация, предназначенная для сокрытия тайных сообщений. Стеганографический канал (стегоканал) — канал передачи контейнера. Одним из примеров стеганографии является способ тайнописи молоком. В компьютерной стеганографии часто используется способ модификации наименее значимых бит, например в файлах формата BMP можно модифицировать 2–3 младших бита в каждом цветовом канале. Для глаза такие картинки будут выглядеть одинаково, однако для посвященных, там будет скрытно передано некоторое сообщение.

Существует большое количество методов внедрения стеганографической информации в различные виды контейнеров: изображения в форматах JPEG, BMP, TIFF и т.д., аудиофайлах, видеофайлах. Программа F5 позволяет внедрять сообщения в изображения формата JPEG.

Задание на лабораторную работу

Исследовать максимальную емкость контейнера в системе стеганографии F5 в зависимости от различных контейнеров.

Порядок действий

1. Подобрать 20 различных картинок (различные размеры типы картинок: фотографии природы, портреты, геометрические картинки, мультипликационные снимки и т.д.).

2. Для каждой картинки внедрить максимальное количество информации.

3. Построить график зависимости объема внедренного сообщения (биты) от исходного объема картинки (биты).

4. Вычислить средний процент объема сообщения к объему контейнера.

5. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.

2. Дата выполнения лабораторной работы.

3. Тема лабораторной работы и ее номер.

4. Все подобранные картинки в уменьшенном варианте.

5. Построенный график зависимости.

6. Вычисленный процент соотношения среднего объема сообщения и среднего объема контейнера.

Контрольные вопросы

1. В какие типы картинок удалось внедрить больше информации?

2. Как долго работает алгоритм F5?

3. Где можно использовать систему F5?

9.6. Псевдослучайные последовательности

В криптографии очень важным является способы построения псевдослучайных последовательностей, поскольку эти последовательности необходимы как для генерации гаммы, так и для получения случайных простых чисел и прочих используемых случайных чисел.

Чтобы получить последовательности элементов гаммы, длина которых превышает размер шифруемых данных, используются датчики псевдослучайных чисел. Существует множество вариантов таких датчиков, наиболее простыми — являются линейные конгруэнтные генераторы псевдослучайных чисел. Они вырабатывают последовательности псевдослучайных чисел x_i, описываемые соотношением

$$x_{i+1} = (A \cdot x_i + C) \bmod m,$$

где A и C — константы, x_0 — исходная величина, выбранная в качестве порождающего числа. Эти три величины и образуют ключ датчика псевдослучайных чисел.

Такой датчик генерирует псевдослучайные числа с определенным периодом повторения, зависящим от выбранных значений A и C. Значение m обычно устанавливается равным 2^n, где n — длина машинного слова в битах. Датчик имеет максимальный период m до того, как генерируемая последовательность начнет повторяться. Следовательно, необходимо выбирать числа A и C такие, чтобы период был максимальным.

Теорема 24. *Линейная конгруэнтная последовательность, определенная числами m, A, C, x_0, имеет период длинной m тогда и только тогда, когда:*

1) числа m и C взаимно просты;

2) $A-1$ кратно p для каждого простого p, являющегося делителем m;

3) $A-1$ кратно 4, если m кратно 4.

Особенностью линейного конгруэнтного генератора является то, что если сомножитель и модуль соответствующим образом подобраны, то результирующая последовательность чисел будет статистически неотличима от случайной последовательности элементов множества $\{0, 1, 2, ..., m-1\}$. Однако, все элементы этой последовательности однозначно определяются четырьмя параметрами: A, C, x_0, m. Если криптоаналитик знает об использовании линейного конгруэнтного метода с известными параметрами, то известной становится вся последовательность чисел. В то же время, даже если криптоаналитик знает только лишь об использовании линейного конгруэнтного метода, то информация о небольшой части последовательности достаточна для выявления параметров метода и всех последующих элементов последовательности. В частности, если криптоаналитику известны значения X_0, X_1, X_2, X_3 , то они удовлетворяют системе уравнений:

$$\begin{cases} X_1 = (A \cdot X_0 + C) \bmod m \\ X_2 = (A \cdot X_1 + C) \bmod m \\ X_3 = (A \cdot X_2 + C) \bmod m \end{cases}$$

из которой можно получить значения параметров A, C, m. Поэтому, хотя линейный конгруэнтный метод порождает статистически хорошую псевдослучайную последовательность чисел, он не является криптографически стойким.

В рамках лабораторной работы будем работать со статистическим пакетом института NIST для генераторов случайных и псевдослучай-

ных чисел в криптографических приложениях. Этот пакет состоит из 15 стандартизированных тестов, целью которых является определение меры случайности двоичных последовательностей. Особенность всех используемых тестов заключается в том, что в результате каждого теста пользователь получает вероятность того, что последовательность случайная. Пакет тестов NIST состоит из следующих тестов:

1. Частотный побитовый тест (Frequency). В рамках теста оценивается соотношение нулей и единиц в битовой последовательности. В случайной последовательности это соотношение близко к 1.

2. Частотный блочный тест (BlockFrequency). В рамках теста оценивается соотношение нулей и единиц в битовых подпоследовательностях длины m. В случайной последовательности количество единиц близко к $m/2$.

3. Тест на последовательность одинаковых битов (Runs). В рамках теста оценивается полное число рядов в исходной последовательности, где под словом «ряд» подразумевается непрерывная подпоследовательность одинаковых битов. Ряд длиной k бит состоит из k абсолютно идентичных битов, начинается и заканчивается с бита, содержащего противоположное значение. Цель данного теста — сделать вывод о том, действительно ли количество рядов, состоящих из единиц и нулей с различными длинами, соответствует их количеству в случайной последовательности.

4. Тест на самую длинную последовательность единиц в блоке (LongestRun). В данном тесте определяется самый длинный ряд единиц внутри блока длиной m бит.

5. Тест рангов бинарных матриц (Rank). В рамках теста производится расчёт рангов непересекающихся подматриц, построенных из исходной двоичной последовательности. Целью этого теста является проверка на линейную зависимость подстрок фиксированной длины, составляющих первоначальную последовательность.

6. Спектральный тест (FFT). В рамках теста оцениваются высоты пиков дискретного преобразования Фурье исходной последовательности. Цель — выявление периодических свойств входной последовательности, например, близко расположенных друг к другу повторяющихся участков. Тем самым это явно демонстрирует отклонения от случайного характера исследуемой последовательности.

7. Тест на совпадение неперекрывающихся шаблонов (NonOverlappingTemplate). В данном тесте подсчитывается количество заранее определенных шаблонов, найденных в исходной последовательности. Цель — выявить генераторы случайных или псевдослучайных чисел, формирующие слишком часто заданные непериодические шаблоны. Как и в тесте № 8 на совпадение перекрывающихся шаблонов для поиска

конкретных шаблонов длиной m бит используется окно также длиной m бит. Если шаблон не обнаружен, окно смещается на один бит. Если же шаблон найден, окно перемещается на бит, следующий за найденным шаблоном, и поиск продолжается дальше.

8. Тест на совпадение перекрывающихся шаблонов (OverlappingTemplate). Суть данного теста заключается в подсчете количества заранее определенных шаблонов, найденных в исходной последовательности. Как и в тесте № 7 на совпадение неперекрывающихся шаблонов для поиска конкретных шаблонов длиной m бит используется окно также длиной m бит. Сам поиск производится аналогичным образом. Если шаблон не обнаружен, окно смещается на один бит. Разница между этим тестом и тестом № 7 заключается лишь в том, что если шаблон найден, окно перемещается только на бит вперед, после чего поиск продолжается дальше.

9. Универсальный статистический тест Маурера (Universal). Здесь определяется число бит между одинаковыми шаблонами в исходной последовательности. Цель теста — выяснить может ли данная последовательность быть значительно сжата без потерь информации.

10. Тест на линейную сложность (LinearComplexity). В основе теста лежит принцип работы линейного регистра сдвига с обратной связью. Цель — выяснить является ли входная последовательность достаточно сложной для того, чтобы считаться абсолютно случайной. Абсолютно случайные последовательности характеризуются длинными линейными регистрами сдвига с обратной связью. Если же такой регистр слишком короткий, то предполагается, что последовательность не является в полной мере случайной.

11. Тест на периодичность (Serial). Данный тест заключается в подсчете частоты всех возможных перекрываний шаблонов длины m бит на протяжении исходной последовательности битов. Целью является определение действительно ли количество появлений $2m$, перекрывающихся шаблонов длиной m бит, приблизительно такое же как в случае абсолютно случайной входной последовательности бит.

12. Тест приблизительной энтропии (ApproximateEntropy). Как и в тесте на периодичность в данном тесте акцент делается на подсчёте частоты всех возможных перекрываний шаблонов длины m бит на протяжении исходной последовательности битов. Цель теста — сравнить частоты перекрывания двух последовательных блоков исходной последовательности с длинами m и $m+1$ с частотами перекрывания аналогичных блоков в абсолютно случайной последовательности.

13. Тест кумулятивных сумм (CumulativeSums). Тест заключается в максимальном отклонении (от нуля) при произвольном обходе, опреде-

ляемым кумулятивной суммой заданных (−1, +1) цифр в последовательности. Цель данного теста — определить является ли кумулятивная сумма частичных последовательностей, возникающих во входной последовательности, слишком большой или слишком маленькой по сравнению с ожидаемым поведением такой суммы для абсолютно случайной входной последовательности. Таким образом, кумулятивная сумма может рассматриваться как произвольный обход.

14. Тест на произвольные отклонения (RandomExcursions). Суть данного теста заключается в подсчёте числа циклов, имеющих строго k посещений при произвольном обходе кумулятивной суммы. Цель данного теста — определить отличается ли число посещений определенного состояния внутри цикла от аналогичного числа в случае абсолютно случайной входной последовательности. Фактически данный тест есть набор, состоящий из восьми тестов, проводимых для каждого из восьми состояний цикла: −4, −3, −2, −1 и +1, +2, +3, +4.

15. Другой тест на произвольные отклонения (RandomExcursionsVariant). В этом тесте подсчитывается общее число посещений определенного состояния при произвольном обходе кумулятивной суммы. Целью является определение отклонений от ожидаемого числа посещений различных состояний при произвольном обходе. В действительности этот тест состоит из 18 тестов, проводимых для каждого состояния: −9, −8, . . . , −1 и +1, +2, . . . , +9.

Задание на лабораторную работу

Протестировать псевдослучайные последовательности, порожденных различными генераторами.

Порядок действий

1. Запустить пакет тестов NIST для встроенного генератора. Генератор выбирается по формуле $g = v \bmod 9 + 1$, где v — номер вариант. Сформировать отчет.

2. Запустить пакет тестов NIST для иррационального числа (иррациональные числа уже вычислены и имеют имя вида data.*). Пусть $n = v \bmod 4$, где v — номер вариант. Если $n = 0$, то в качестве числа взять число π, если $n = 1$, то e, если $n = 2$, то $\sqrt{2}$, если $n = 3$, то $\sqrt{3}$. Сформировать отчет.

3. Запустить пакет тестов NIST для текстового файла.Сформировать отчет.

4. Запустить пакет тестов NIST для исполнимого файла формата EXE. Сформировать отчет.

5. Сравнить отчеты. Указать слабые места построенные последовательностей. Сделать выводы о криптостойкости.

6. Подготовить, сдать и защитить отчет.

Содержание отчет

1. Номер бригады и её состав с указанием полных ФИО, номера группы.

2. Дата выполнения лабораторной работы.

3. Тема лабораторной работы и ее номер.

4. Четыре отчета.

5. Таблица сравнения генераторов.

6. Выводы о практической целесообразности использования генераторов псевдослучайных чисел.

Контрольные вопросы

1. Какие есть способы получения псевдослучайных чисел?

2. Какие есть способы получения случайных чисел?

3. Какой генератор из рассмотренных оказался лучше и почему?

9.7. Изучение лавинного эффекта в симметричных алгоритмах шифрования

Общие требования к узлам замен (S-блокам) блочных шифров повторяют требования к функции шифрования в целом – это нелинейность и лавинный эффект. В идеале любые изменения входных данных узла должны приводить к «случайным» изменениям выходных данных (если рассматривать узел замен как «черный ящик»). Существует ряд общеизвестных критериев для проектирования устойчивых к дифференциальному криптоанализу узлов замен для любых блочных шифров.

1. Строгий критерий лавинного эффекта – требует, чтобы для любых i и j при инвертировании входного бита i на входе узла замен выходной бит j изменялся с вероятностью 0.5.

2. Критерий независимости битов – требует, чтобы для любых значений i, j и k при инвертировании входного бита i на входе узла замен выходные биты j и k изменялись независимо (вероятность одновременного изменения битов должна быть равна произведению вероятностей изменения отдельных бит).

3. Критерий гарантированного лавинного эффекта порядка γ – выполняется, если при изменении одного бита на входе узла замен на выходе меняются как минимум γ выходных битов.

Задание на лабораторную работу

Реализовать алгоритм шифрования ГОСТ 28147-89 (режим простой замены, S-блоки Центрального банка РФ) и исследовать лавинный эффект (по исходным данным и по ключу) в зависимости от количества раундов в алгоритме шифрования.

Порядок действий

1. Реализовать алгоритм шифрования ГОСТ 28147-89.
2. Построить график зависимости среднего количества бит выходного вектора, изменяющихся при изменении одного бита входного вектора в зависимости от количества раундов в алгоритме (от одного до 32). Усреднение вести по 64 изменяемым входным битам. Исходные входные биты выбрать однократно и случайно.
3. Построить график зависимости среднего количества бит выходного вектора, изменяющихся при изменении одного бита ключего вектора в зависимости от количества раундов в алгоритме (от одного до 32). Усреднение вести по 256 изменяемым входным битам. Исходные ключевые биты выбрать однократно и случайно.
4. Построить график зависимости минимального количества бит выходного вектора, изменяющихся при изменении одного бита входного вектора в зависимости от количества раундов в алгоритме (от одного до 32). Агрегацию вести по 64 изменяемым входным битам. Исходные входные биты выбрать однократно и случайно.
5. Построить график зависимости минимального количества бит выходного вектора, изменяющихся при изменении одного бита ключего вектора в зависимости от количества раундов в алгоритме (от одного до 32). Агрегацию вести по 256 изменяемым входным битам. Исходные ключевые биты выбрать однократно и случайно.
6. Сделать выводы о выполнимости критериев лавинного эффекта в зависимости от количества раундов.
7. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Все построенные графики.
5. Выводы о выполнимости критериев лавинного эффекта.
6. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Что обеспечивает лавинный эффект?
2. Каков максимальный уровень лавинного эффекта?
3. Как проверить критерий независимости битов?

9.8. Исследование коллизий хэш-функции

Определение 30. *Хэш-функция* — это преобразование по определённому алгоритму входных данных произвольной длины в выходную битовую строку фиксированной длины.

Определение 31. *Криптографическая хэш-функция* — это такая хэш-функция, которая необратима и устойчива к коллизиям первого и второго рода.

Необратимость означает, что для заданного значения хэш-функции m должно быть вычислительно невозможно найти блок данных X, для которого $H(X) = m$.

Стойкость к коллизиям первого рода означает, что для заданного сообщения M должно быть вычислительно невозможно подобрать другое сообщение N, для которого $H(N) = H(M)$.

Стойкость к коллизиям второго рода означает, что должно быть вычислительно невозможно подобрать пару сообщений (M, M'), имеющих одинаковый хэш.

Обычно сами пароли не хранятся в системе, вместо паролей хранятся их хэши. В этом случае, если у двух пользователей одинаковый пароль, то при утечке файла с хэшами это может быть сразу же обнаружено. Для того, чтобы это скрыть (и заодно усложнить перебор) используется соль.

Определение 32. *Соль* — это строка случайных данных, которая подается на вход хэш-функции вместе с исходными данными.

Сама соль сохраняется в открытом виде рядом с хэш-значением. И при проверке пароля к паролю дописывается соль, получившаяся строка подается на вход хэш-функции, а результат сверяется с базой хэшей.

Задание на лабораторную работу

Исследовать стойкость к коллизиям первого и второго рода и сравнить с теоретически ожидаемым.

Порядок действий

1. Построить хэш-функцию на основе алгоритма SHA-1. Хэш-функция должна возвращать только первые k бит из возвращаемых 160 бит.

2. Сформировать два сообщения: “Прошу перевести 8000 руб 00 коп на счет №01010101. Клиент <CASE>. RAND=<SALT>” (вид 1) и и “Прошу перевести 8000 руб 00 коп на счет №01010102. Клиент <CASE>. RAND=<SALT>” (вид 2), где вместо <CASE> подставить двузначный номер свой бригады с ведущим нулем, если требуется, а вместо <SALT> — подставить 128-байтную соль.

3. Перебирая все k от 1 до 24 бит, для каждого k найти необходимое количество случайно сгенерированных сообщений вида 2 до тех пор, пока k-битный хэш от сообщения вида 1, сформированного на втором шаге, не совпадет с первым сообщением вида 2, сгенерированным на третьем шаге.

4. Перебирая все k от 1 до 48 бит, для каждого k найти необходимое количество случайно сгенерированных сообщений вида 1 и вида 2 до тех пор, пока k-битный хэш от сообщения вида 1 не совпадет с любым хэш-значением сообщения вида 2, сгенерированным на этом шаге ранее, либо пока k-битный хэш от сообщения вида 2 не совпадет с любым хэш-значением сообщения вида 2, сгенерированным на этом шаге ранее.

5. Для каждого k теоретическую оценку трудоемкости.

6. Сделать выводы о зависимости криптостойкости хэш-функции от ее длины.

7. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.

2. Дата выполнения лабораторной работы

3. Тема лабораторной работы и ее номер.

4. Диаграмму зависимости необходимого количества сгенерированных сообщений от k при тестировании коллизии первого рода. На диаграмме показать теоретическую оценку.

5. Диаграмму зависимости необходимого количества сгенерированных сообщений от k при тестировании коллизии второго рода. На диаграмме показать теоретическую оценку.

6. Выводы о зависимости криптостойкости хэш-функции от ее длины.

7. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Как можно использовать коллизии первого рода?
2. Как можно использовать коллизии второго рода?
3. Какова криптостокость хэш-функции длинной n бит?

9.9. Изучение распределения простых чисел

Задание на лабораторную работу

Изучить плотность распределения простых чисел, сравнить эффективность алгоритма Миллера—Рабина, теста Ферма и теста пробных делений.

Порядок действий

1. Равномерно сгенерировать 100 000 чисел из 9 цифр. Построить гистограмму (в логарифмическом масштабе) распределения количества значимых цифр в получившихся числах.
2. Для каждого числа применить алгоритм Миллера—Рабина (с одним основанием), тест Ферма (с одним основанием), алгоритм Миллера—Рабина (с двумя основаниями), тест Ферма (с двумя основаниями), тест пробных делений. Вычислить среднее время работы каждого алгоритма и вероятность ошибки.
3. Изучить плотность распределения простых чисел, построить график и сравнить с теоретически ожидаемым.
4. Сделать выводы об эффективности алгоритмов.
5. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Гистограмма распределения сколько значные числа будут использоваться.
5. Результаты тестирования алгоритмов проверки на простоту.
6. Таблица плотности распределения простых чисел, теоретические границы.
7. Выводы о практической целесообразности использования алгоритмов поиска больших простых чисел.
8. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Какой из методов оказался самым быстрым? Самым точным?
2. Когда ошибается тест Ферма? Помогает ли второе основание?

9.10. Вычисление символов Лежандра

Задание на лабораторную работу

Вычислить символы Лежандра и раскодировать сообщение.

Порядок действий

1. Вычислить заданные символы Лежандра.
2. Учитывая, что исходный текст — текст на русском языке и при кодировании нулевой бит заменялся квадратичным невычетом, а единичный — квадратичным вычетом, раскодировать исходное сообщение.
3. Подготовить, сдать и защитить отчет.

Содержание отчет

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Вычисленные значения символов Лежандра.
5. Раскодированное сообщение.
6. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Каким образом вычислялся символ Лежандра? Есть ли альтернативные способы?

9.11. Использование системы GPG

Gnupg GnuPG (GNU Privacy Guard, «Страж безопасности GNU», или просто GPG) – это открытый эквивалент PGP (Pretty Good Privacy), известной и широкоиспользуемой программы для Windows, DOS и других операционных систем. Он распространяется в открытых исходниках и имеет те же самые функции, что и PGP, основанные на стандарте OpenPGP.

Задание на лабораторную работу

Изучить способы работы с программой GPG.

Порядок действий

1. Для проведения работы выбрать два компьютера. Один член бригады работает на одном компьютере, а другой на другом. В отчет включить описание компьютеров, скриншоты рабочих столов и результаты вывода команды `msinfo32`. Выбрать псевдонимы для компьютеров.

2. На первом компьютере создать текстовый файл большого объема (несколько мегабайтов). Зашифровать его с помощью алгоритма симметричного шифрования на парольной фразе с помощью команды `gpg --symmetric имя_файла`.

3. Передать зашифрованный файл на второй компьютер и расшифровать этот файл с помощью команды `gpg --decrypt имя_файла.gpg -o имя_файла2`. Убедиться, что имя_файла и имя_файла2 идентичны. Проверить идентичность исходного файла и расшифрованного (как?).

4. Сгенерировать пару ключей для асимметричного шифрования с помощью команды `gpg --gen-key` на каждом компьютере.

5. Посмотреть список ключей с помощью команды `gpg --list-keys` на каждом компьютере.

6. Создать сертификат отзыва на случай компрометации ключа или его утраты с помощью команды `gpg --gen-revoke ID ключа` на каждом компьютере.

7. Экспортировать свой публичный ключ с помощью команды `gpg --export --armor ID ключа`.

8. Обменяться публичными ключами.

9. Импортировать полученные публичные ключи с помощью команды `gpg --import имя.gpg`.

10. Посмотреть отпечаток полученного ключа с помощью команды `gpg --fingerprint ID ключа` и удостовериться, что при копировании ключ не был подменён.

11. Посмотреть список подписей на ключах с помощью команды `gpg --list-sigs`.

12. Подписать полученный ключ с помощью команды `gpg --sign-key ID ключа`.

13. Экспортировать подписанный публичный ключ другому компьютеру.

14. Установить степень доверия ключу командой `gpg --edit-key ID ключа`.

15. Создать текстовое сообщение большого объема (несколько мегабайт) и асимметрично зашифровать его на публичном ключе другого компьютера с помощью команды `gpg --encrypt message.txt --recipient ID ключа`.

16. Отправить зашифрованное сообщение на другой компьютер.

17. Получив зашифрованное сообщение, расшифровать его с помощью команды `gpg --decrypt message.gpg --output message.txt`.

18. Создать текстовое сообщение большого объема и подписать его электронно-цифровой подписью с помощью команды `gpg --clearsign message.txt`.

19. Отправить подписанное сообщение на другой компьютер.

20. Получив подписанное сообщение, проверить его подпись с помощью команды `gpg --verify message.asc`.

Содержание отчет

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Снимки экрана после каждого выполненного действия.
5. Содержание полученных или использованных файлов.
6. Выводы об удобстве использования программы.

Контрольные вопросы

1. Как зашифровать файл с помощью GPG?
2. Как подписать файл с помощью GPG?
3. Как сгенерировать ключи с помощью GPG?

9.12. Криптоанализ алгоритма RSA

Задание на лабораторную работу

Прочитать сообщение, зашифрованное алгоритмом RSA при известном открытом ключе.

Порядок действий

1. Разложить модуль n на множители p, q с помощью ρ-алгоритма Полларда.

2. Вычислить $\varphi(n)$ и с помощью расширенного алгоритма Евклида определить закрытый ключ d.

3. Расшифровать каждый широтекст и раскодировать каждое исходное сообщение в тройку символов ASCII с учетом того, что при кодировании использовалась формула $M_i = c_1 \cdot 65536 + c_2 \cdot 256 + c_3$, где c_1, c_2, c_3 — первый, второй и третий символ соответственно. Исходный текст — текст на русском языке.

4. Подготовить, сдать и защитить отчет.

Содержание отчета

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Найденные значения $p, q, \varphi(n), d$.
5. Полученные исходные сообщения и раскодированный текст.
6. Выводы о самом сложном этапе в криптоанализе RSA.
7. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Каким образом были получены p, q?
2. Каким образом было получено $\varphi(n)$?
3. Каким образом было получено d?

9.13. Дискретное логарифмирование

Задание на лабораторную работу

Провести криптоанализ шифротекста зашифрованного алгоритмом Эль-Гамаля.

Порядок действий

1. С помощью алгоритма больших и малых шагов найти x.
2. Расшифровать шифротекст.
3. Подготовить, сдать и защитить отчет.

Содержание отчет

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Исходная шифрограмма и открытый ключ.
5. Совпадающие значения в таблицах в алгоритме больших и малых шагов.
6. Результат дешифровки.
7. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Какие особенности есть у алгоритма больших и малых шагов?
2. В чем основная сложность криптоанализа этого алгоритма?

9.14. Криптоанализ алгоритма Рабина

Задание на лабораторную работу

Расшифровать шифротекст, зашифрованный криптосистемой Рабина, при условии, что исходный текст является текстом на русском языке.

Порядок действий

1. Разложить открытый ключ n на простые множители p и q.
2. Извлечь из каждого числа C_i квадратный корень по модулю $p \cdot q$.
3. Учитывая, что исходное сообщение — текст на русском языке и при кодировании использовалась формула $M_i = c_1 \cdot 65536 + c_2 \cdot 256 + c_3$, где c_1, c_2, c_3 — первый, второй и третий символ соответственно, выбрать подходящие корни и раскодировать исходное сообщение.
4. Сделать выводы о самом трудоемком этапе криптоанализа алгоритма.
5. Подготовить, сдать и защитить отчет.

Содержание отчет

1. Номер бригады и её состав с указанием полных ФИО, номера группы.
2. Дата выполнения лабораторной работы.
3. Тема лабораторной работы и ее номер.
4. Данное n, полученные p, q. Вычисленные четверки квадратных корней.
5. Описание процесса поиска шифротекста с поэтапным представлением результатов.
6. Выводы о самом трудоемком этапе криптоанализа алгоритма.
7. Если была разработана программа, то ее листинг.

Контрольные вопросы

1. Какой метод использовался для поиска p, q?
2. Как выбиралось M_i?

10. КОНТРОЛЬНЫЕ ВОПРОСЫ

Основы защиты информации.

1. Современная компьютерная система.
2. Исторические подходы к защите информации.
3. Принципы защиты информации.
4. Общая характеристика средств и методов защиты информации в компьютерных системах.
5. Стандартные атаки. Полный перебор.

Классические шифры (оценки криптостойкости).

1. Исторические этапы развития криптографии.
2. Шифр Цезаря.
3. Полибианский квадрат.
4. Одиночная перестановка по ключу.
5. Шифрование методом двойной перестановки.
6. Магические квадраты.
7. Шифр перестановки «скитала».
8. Шифр Виженера.
9. Таблица Трисемуса.
10. Шифр Плейфера.
11. Шифр двойного квадрата.
12. Система омофонов.
13. Шифр Хилла.
14. Роторные машины.

Криптосистемы с закрытым ключом.

1. Моно- и полиалфавитные подстановки.
2. Перестановки.
3. Гаммирование. Датчики псевдослучайных чисел.
4. Абсолютная криптостойкость.
5. Сеть Фейстеля. SP-сеть.
6. Алгоритм ГОСТ 28147-89. Режимы шифрования.

Криптосистемы с открытыми ключами

1. Алгоритм Меркла-Хеллмана.
2. Алгоритм Рабина.
3. Алгоритм RSA.
4. Алгоритм шифрования Эль-Гамаля.
5. Алгоритм Диффи-Хеллмана.
6. Алгоритм Диффи-Хеллмана для эллиптических кривых.
7. Алгоритм шифрования на основе эллиптических кривых.

8. Алгоритм ГОСТ Р 34.10-2012.

Математические основы криптографии

1. Алгоритм Евклида. Расширенный алгоритм Евклида.
2. Линейный диофантовы уравнения.
3. Группы. Теорема Лагранжа. Кольца. Поля.
4. Вычеты. Обратные элементы по сложению и умножению.
5. Линейные сравнения с одним неизвестным.
6. Простые числа. Решето Эратосфена.
7. Алгоритм быстрого возведения числа в степень по модулю.
8. Малая теорема Ферма.
9. Теорема Эйлера.
10. Проверка числа на простоту. Алгоритм AKS. Тест Ферма. Тест квадратным корнем.
11. Алгоритм Миллера—Рабина.
12. Основная теорема арифметики. Метод проверки делением. Метод Ферма.
13. $P-1$ метод Полларда. ρ-метод Полларда.
14. Китайская теорема об остатках.
15. Сравнения второй степени по простому модулю. Критерий Эйлера. Символ Лежандра.
16. Дискретный логарифм. Первообразные корни.
17. Алгоритм больших и малых шагов.
18. Алгоритм дискретного логарифмирования Полига-Хеллмана.
19. Эллиптические кривые.
20. Эллиптические кривые в конечных полях.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

алгоритм AKS, 78
алгоритм P-1 Полларда, 83
алгоритм RSA, 99
алгоритм Диффи-Хеллмана, 93
алгоритм Диффи-Хеллмана на основе эллиптической кривой, 95
алгоритм Евклида, 38
алгоритм Евклида расширенный, 39
алгоритм Эль-Гамаля, 101
алгоритм Ферма разложения на множители, 82
алгоритм Миллера—Рабина, 74
алгоритм Полига-Хеллмана, 65
алгоритм Рабина, 103
алгоритм больших и малых шагов, 64
алгоритм пробных делений, 72
алгоритм проверки делением, 81
алгоритм рюкзака, 95
алгоритм шифрования асимметричный, 93
алгоритм шифрования на основе эллиптических кривых, 104
алгоритм шифрования симметричный, 27
алгоритм ρ Полларда, 84
алгоритма ГОСТ 28147-89, 30
атака человек посередине, 21
атака по сторонним каналам, 22
атака, полный перебор, 25
аутентичность, 4
целостность, 4
числа Ферма, 72
числа Мерсенна, 71
числа взаимно простые, 34
число Кармайкла, 55
число простое, 34
число составное, 34
доступность, 4
эцп на основе эллиптических кривых, 105
элемента порядок, 46
функция распределения простых чисел, 69
группа, 44
группа абелева, 45
группа циклическая, 46
группы порядок, 46
характеристика поля, 87
хэш-функция криптографическая, 128
класс квадратичных невычетов, 57
класс квадратичных вычетов, 57
класс вычетов, 35
кольцо, 45
конфиденциальность, 4
криптоанализ, алгебраический, 21
криптоанализ, дифференциальный, 21
криптоанализ, линейный, 21
криптоанализ, статистический, 21
криптография, этапы, 9
криптосистема, абсолютно стойкая, 20
криптосистема, безусловно стойкая, 18
криптосистема, вычислительно стойкая, 18
критерий Эйлера, 58
кривая эллиптическая, 87
лавинный эффект, 28
лавинный эффект, диффузия, 28

лавинный эффект, конфузия, 28
лемма Безу, 39
методы защиты информации, 5
модель нарушителя, 17
наибольший общий делитель, 37
неотказуемость, 4
обратный элемент, 42
отношение сравнения, 34
парадокс дней рождения, 24
первообразный корень, 63
подгруппа, 46
подгруппа циклическая, 46
показатель криптостойкости, 17
поле, 45
порядок точки на эллиптической кривой, 89
последовательность линейная конгруэнтная, 122
последовательность сверхвозрастающая, 97
принципы защиты информации, 6
решето Эратосфена, 69
режимы применения ГОСТ 28147-89, 31
сеть SP, 28
сеть Фейстеля, 29
символ Лежандра, 60
система информационная, 3
сравнение линейное, 47
сравнение второй степени, 56
стегоконтейнер, 120
шифр Цезаря, 9
шифр Хилла, 14
шифр Плейфера, 13
шифр Виженера, 13
шифр двойным квадратом, 13
шифр двойной перестановки, 11
шифр магический квадрат, 11
шифр моноалфавитный, 113
шифр одиночной перестановки, 10
шифр омофонов, 14
шифр полиалфавитный, 115
шифр полибианский квадрат, 10
шифр роторной машины, 16
шифр скитала, 12
таблица Трисемуса, 13
технология информационная, 3
теорема AKS, 78
теорема Эйлера, 55
теорема Ферма малая, 53
теорема Хассе, 92
теорема Лагранжа., 46
теорема арифметики, основная, 80
теорема китайская об остатках, 48
теорема о делении с остатком, 33
теорема о количестве первообразных корней, 63
теорема о количестве простых чисел, 69
теорема о мультипликативности функции Эйлера, 50
теорема о периоде линейной конгруэнтной последовательности, 122
теорема о представителях квадратичных вычетов, 58
теорема о существовании первообразных корней, 63
теорема об абсолютно стойком шифре, 20
теорема об эквивалентности квадратичных сравнений, 56
тест Ферма, 73
тест Люка-Лемера, 71
тест Пепина, 72
тест квадратного корня, 74
требование Керкхоффа, 16
уравнение линейное диофантово, 41

P-блок, 28

S-блок, 28

$\varphi(n)$-функция Эйлера, 49

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ван-дер-Варден, Б. Л. Алгебра / Б. Л. Ван-дер-Варден. — 2-изд. — М.: Наука, 1979.
2. Василенко, О. Н. Теоретико-числовые алгоритмы в криптографии / О. Н. Василенко. — М.: МЦНМО, 2003.
3. Коробейников, А. Г. Математические основы криптологии. Учебное пособие / А. Г. Коробейников, Ю.А. Гатчин. — СПб: СПб ГУ ИТМО, 2004.
4. Введение в криптографию / под общ. ред. В. В. Ященко. — 4-е изд., доп. М.: МЦНМО, 2012.
5. Ишмухаметов, Ш.Т. Методы факторизации натуральных чисел / Ш.Т. Ишмухаметов. — Казань, 2012.
6. Коблиц, Н. Курс теории чисел и криптографии / Н. Коблиц. — М.: ТВП, 2001.
7. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. — М.: МЦНМО, 1999.
8. Криптографическая защита информации: учебное пособие / А.В. Яковлев, А.А. Безбогов, В.В. Родин, В.Н. Шамкин. — Тамбов : Изд-во Тамб. гос. техн. ун-та, 2006.
9. Столлингс, В. Криптография и защита сетей: принципы и практика / В. Столлингс. — М.: Вильямс, 2001.
10. Черемушкин, А . В. Криптографические протоколы. Основные свойства и уязвимости: учебное пособие / А. В. Черемушкин — М .: Издательский центр «Академия», 2009.
11. Чмора, А.Л. Современная прикладная криптография. — 2-е изд. / А. Л. Чмора. — М.: Гелиос, АРВ, 2002.
12. Шаньгин, Ф.Ф. Защита компьютерной информации: эффективные методы и средства / Ф.Ф. Шаньгин. — М.:ДМК, 2008.
13. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. — М.: Триумф, 2002.
14. Элементарные введение в эллиптическую криптографию: Алгебраические и алгоритмические основы/ А. А. Болотов, С. Б. Гашков, А. Б. Фролов, А. А. Часовских — М.:КомКнига, 2006.
15. Ященко, В. В. Введение в криптографию / В. В. Ященко. — СПб.: Питер, 2001.

ОГЛАВЛЕНИЕ

1. ЗАЩИТА ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ 3
1.1. Общая характеристика средств и методов защиты информации в компьютерных системах 5
1.2. Основные принципы защиты информации 6

2. КЛАССИЧЕСКИЕ ШИФРЫ 9
2.1. Криптостойкость 17
2.2. Криптоанализ 21
2.3. Уязвимости организации процесса передачи сообщений . . . 21
2.4. Парадокс дней рождения 24
2.5. Полный перебор 25

3. СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ 27
3.1. Подстановочно-перестановочная сеть 28
3.2. Сеть Фейстеля 29
3.3. Стандарт шифрования ГОСТ 28147-89 30

4. ВВЕДЕНИЕ В ТЕОРИЮ ЧИСЕЛ 33
4.1. Модульная арифметика 34
4.2. Наибольший общий делитель 37
4.3. Линейные диофантовы уравнения 41
4.4. Обратные элементы 42
4.5. Группы 44
4.6. Линейное сравнение 47
4.7. Системы линейных сравнений 48
4.8. φ−функция Эйлера 49
4.9. Алгоритм быстрого возведения в степень по модулю 51
4.10. Малая теорема Ферма 53
4.11. Теорема Эйлера 55
4.12. Квадратичные сравнения 56
4.13. Символ Лежандра 60
4.14. Возведение в степень и логарифмы 62

5. ПРОСТЫЕ ЧИСЛА 69
5.1. Генерация простых чисел 71
5.2. Испытание простоты чисел 72
5.3. Разложение на множители 80

6. ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ 87
6.1. Группа точек эллиптической кривой 87
6.2. Эллиптические кривые над конечными полями 90

7. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ 93
7.1. Алгоритм обмена ключами Диффи-Хеллмана 93
7.2. Алгоритм обмена ключами Диффи-Хеллмана на основе эллиптической кривой 95
7.3. Алгоритм рюкзака для шифрования 95
7.4. Алгоритм шифрования RSA 99
7.5. Алгоритм шифрования Эль-Гамаля 101
7.6. Алгоритм шифрования Рабина 103
7.7. Алгоритм шифрования на основе эллиптических кривых . . 104
7.8. Электронная цифровая подпись на основе эллиптических кривых 105

8. ПРАКТИЧЕСКИЕ ЗАДАНИЯ 109

9. ЛАБОРАТОРНЫЕ РАБОТЫ 113
9.1. Криптоанализ моноалфавитного шифра 113
9.2. Криптоанализ полиалфавитного шифра 115
9.3. Криптоанализ методом вероятных слов 117
9.4. Криптоанализ методом полного перебора 118
9.5. Стеганография 120
9.6. Псевдослучайные последовательности 121
9.7. Изучение лавинного эффекта в симметричных алгоритмах шифрования 126
9.8. Исследование коллизий хэш-функции 128
9.9. Изучение распределения простых чисел 130
9.10. Вычисление символов Лежандра 131
9.11. Использование системы GPG 131
9.12. Криптоанализ алгоритма RSA 133
9.13. Дискретное логарифмирование 134
9.14. Криптоанализ алгоритма Рабина 135

10. КОНТРОЛЬНЫЕ ВОПРОСЫ 136

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ 138

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 140